User manual EWB32 for Windows

Book E34 - Version 1.0.04

E.D.&A. nv
Franseweg 20, 2920 Kalmthout, Belgium • Tel +32 3 620 18 18 • info@edna.eu • www.edna.eu • RPR Antwerp • VAT BE 0459.777.129
KBC • IBAN BE91 4045 0819 5176 • BIC KREDBEBB • BNP Paribas Fortis • IBAN BE13 0016 7093 5639 • BIC GEBABEBB

Table of contents

1 System requirements and compatibility

EWB32 requires a PC Pentium model with at least 32/64 Mbyte RAM.
As operating system:

- WindowsTM ME,
- WindowsTM 2000
- WindowsTM XP

2 Installation of EWB32 and parts

Installing EWB32 is very easy. To install the program you proceed as follows:

1. Place the EWB32 CD-ROM in the suited station.
2. Select the CD-ROM station and double click on the Setup.exe file to start the installation program.
3. The installation program guides you through various elements of the installation process. Follow the instructions on the screen and read all the information the installation program shows you. Then click on the "Continue" button to proceed with the installation.

3 Get busy quickly

3.1 What and how

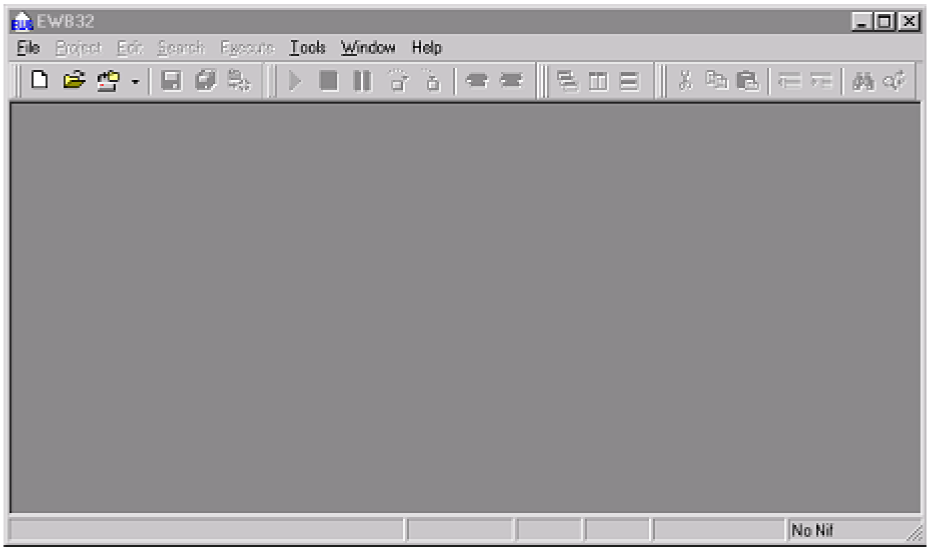
If you cannot wait to begin, we definitely advise you to read this chapter! By means of clear pictures, the different steps that are necessary for the development of EPL programs with EWB32 are shown. If we go through these steps together, you will notice that at the end of this chapter you will be able to create a whole new project.

3.2 Introduction of the HPC

Consult the Hardware Reference Manual of the HPC concerning the introduction of the appliance. Connect the HPC and the PC by means of the suited programming cable as prescribed in the Hardware Reference Manual. Make sure that you carefully read the chapter concerning safety. It contains important guidelines both for the machine and for the application of HPCs in an industrial environment.

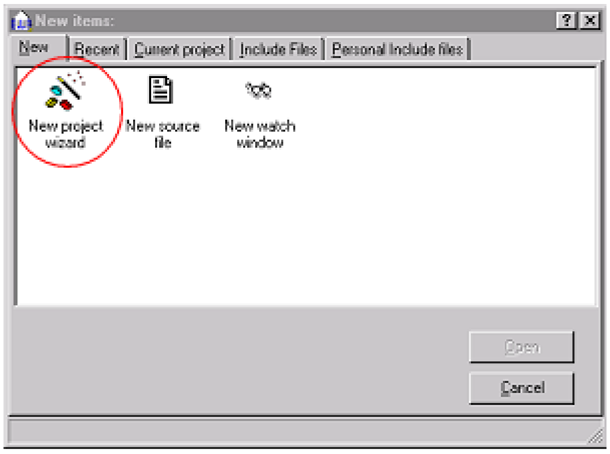
3.3 Step by step

3.3.1 Step 1: Starting up EWB32



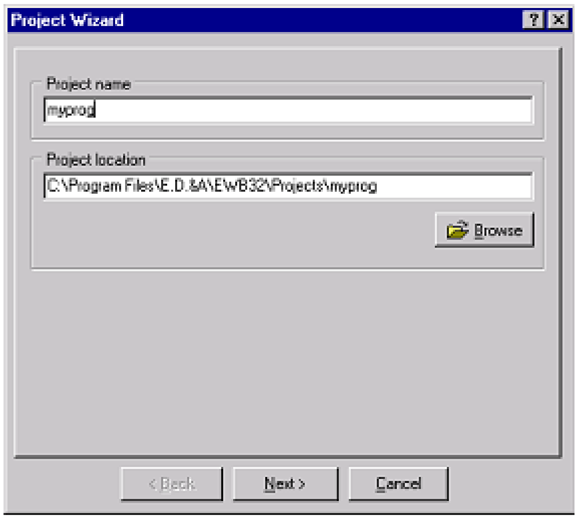
After you have followed the guidelines of the installation program, you can start up the programming environment by executing the short cut Start | Programs | ewb32. After loading the programming environment you will see the screen alongside. EWB32 can be setup to open the project that was open when EWB32 was closed off the previous time. Because it is the first time that EWB32 is being loaded, no project will be opened.

3.3.2 Step 2: New project



In this session we prefer to create a new project. Select File | New… in the menu.
The “New project wizard” module will guide us while we are creating a new project. Select the “New project wizard” button by going towards it with the mouse and clicking on it with the left mouse button.
Subsequently use the left mouse button to click on <Open>.

3.3.3 Step 3: Enter the project name

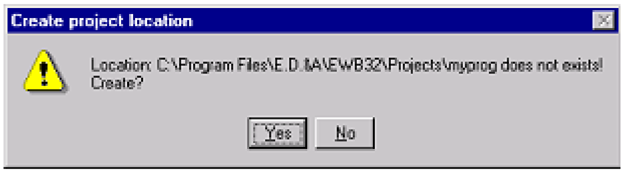


Project name: myprog

Project location: C:\Program Files\E.D.&A\EWB32\Projects\myprog

We choose a suitable name for our project. Select the input field Project name by moving the mouse towards it and pressing the left mouse button. Enter the name myprog as the new project name.
The program also suggests creating a new directory by the same name. Make a habit of it to give each project its own directory.
Then click on <Next> to go to the next screen.

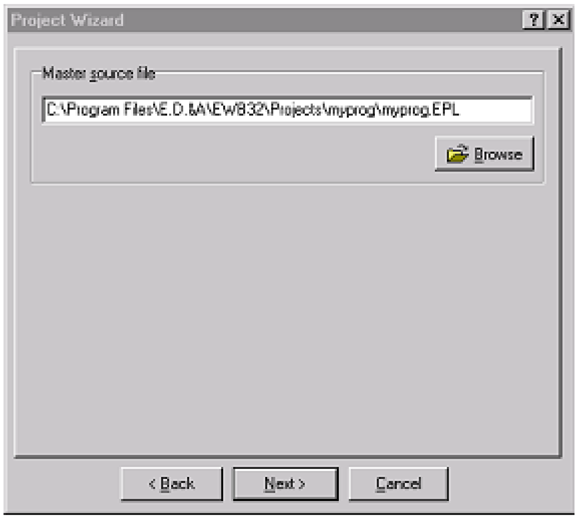
3.3.4 Step 4: Confirm the creation of the new directory



Location: C:\Program Files\E.D.&A\EWB32\Projects\myprog does not exists!
Create?

Because we have chosen a name for our program that did not yet existed on the hard disk, the system asks for a confirmation for creating this project. Check whether the indicated project corresponds with the name we have chosen and then press <Yes>.

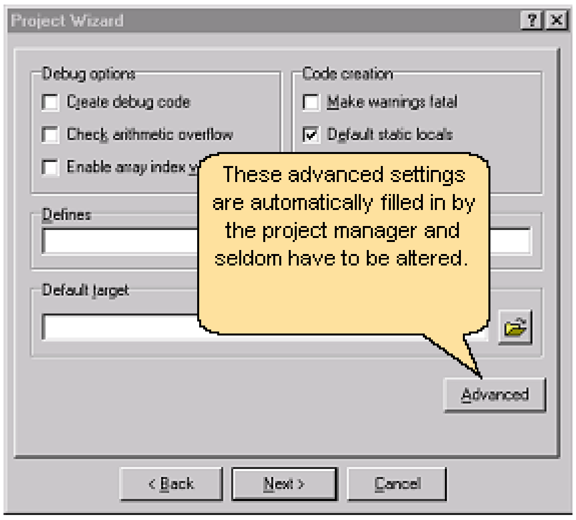
3.3.5 Step 5: Choose the master source file



C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.EPL

Then we can select the file we wish to use as master source file. If we wish to use of an existing file, we can select this file in the Browse dialog. In this training session we do not use an existing file. We accept the file name myprog. An EPL source file automatically receives the extension .epl. Then we press <Next> to continue.

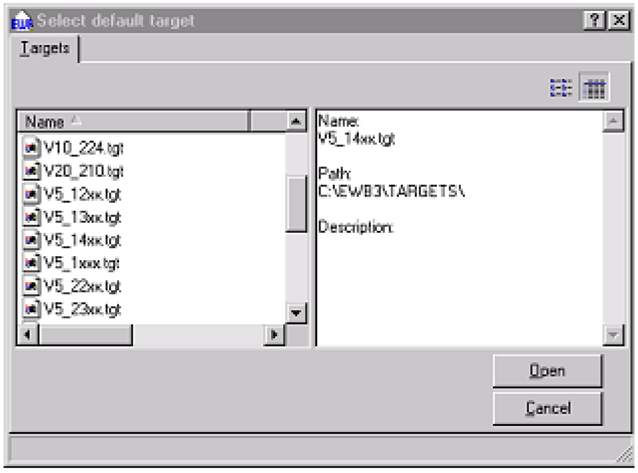
3.3.6 Step 6: Settings of the translator


Project Wizard
Debug options
Code creation
Defines
Default target
Advanced
< Back
Next >
Cancel
These advanced settings are automatically filled in by the project manager and seldom have to be altered.

- [ ] Create debug code
- [ ] Check arithmetic overflow
- [ ] Enable array index
- [ ] Make warnings fatal
- [x] Default static locals

The next window that appears is that of the options for the compiler. The settings in this window influence the translation of an EPL source program into feasible instructions for the HPC. To help the compiler with the translation of the source program, it is necessary that you indicate for which type of HPC you want to develop a program. This is called the 'Target' indication. Click on the map button to set the 'target' HPC.

3.3.7 Step 7: Select a target



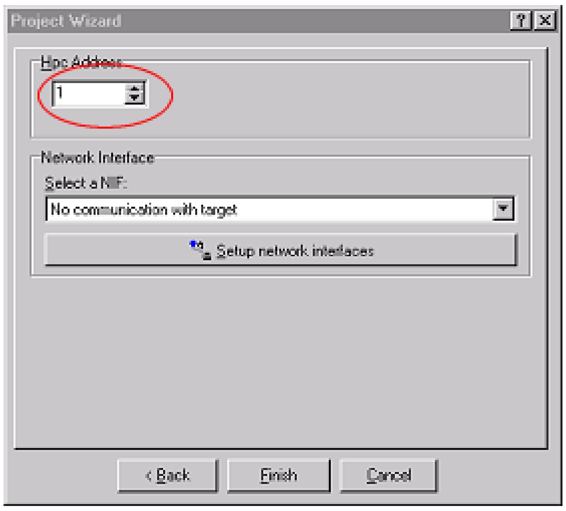
Name:
V5_14xx.tgt

Path:
C:\EWB3\TARGETS\

Description:

The project management opens a window containing the list of installed HPCs for which you can develop a program. The number of different HPC models you see on this list depends on the number that is installed on your system. The indication of a 'target' or a HPC model corresponds with the order number of the CPU card of the HPC or with the order number of the base model. Click on the 'target' indication that corresponds with the HPC model for which you wish to develop a program and press <OK> to continue.

3.3.8 Step 8: Set the remote HPC address

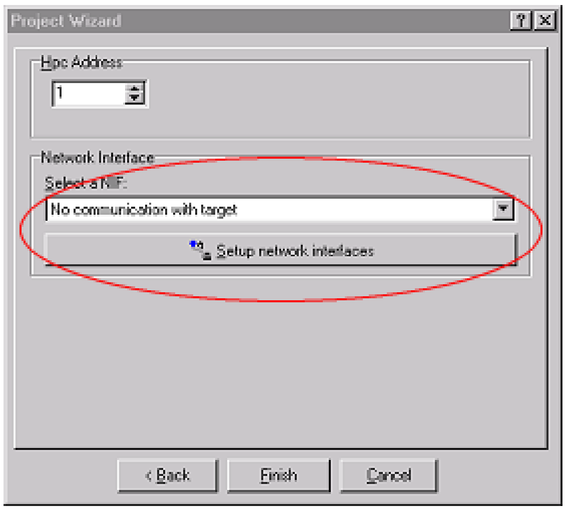


Hpc Address: 1

Network Interface
Select a NIF: No communication with target

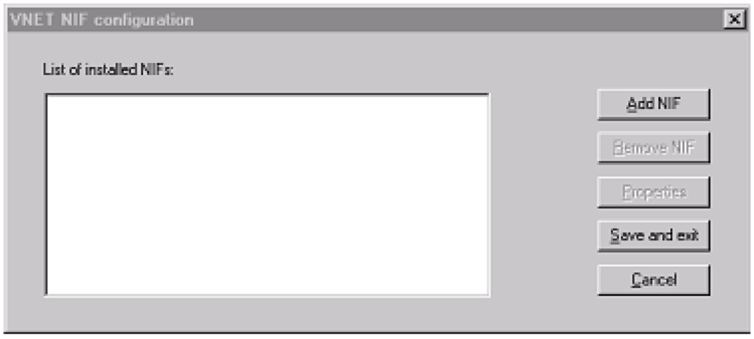
Then you see the dialog window concerning the communication. In this window you set all the things concerning the communication with the HPC. Make sure that the value in the input field of the HPC address corresponds with the HPC address that is set on your HPC. For setting the address of your HPC, we refer to the Hardware Reference Manual of your machine. The standard value for this setting is 1.

3.3.9 Step 9: Select a Nif



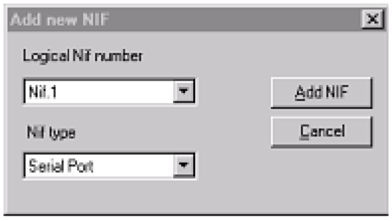
Then choose the port that will be used to communicate with the HPC. Choose from the selection list. Click with the left mouse button on the arrow at the back of the input field. If this list only contains the "No communication with target" choice, you can use the <Setup network interface> button to create an interface.
In the next steps we go through the creation of a NIF. You can skip these steps if you have already created NIFs.

3.3.10 Step 10: Create a network interface



In the list "List of installed NIFs" you see all the interfaces you have already created on your system. But in this example no NIFs have been created. To add a NIF, use the <Add NIF> button.

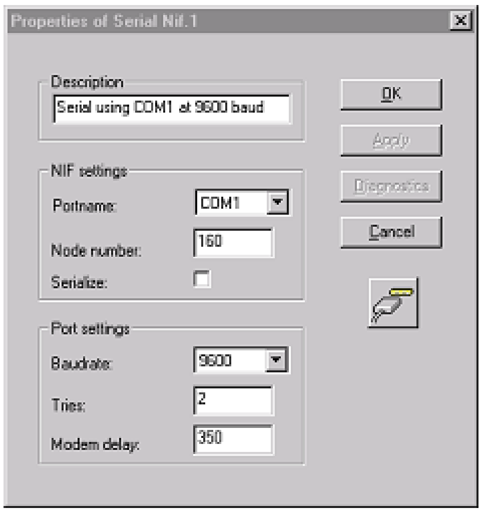
3.3.11 Step 11: Add a new NIF



Choose a Logical NIF number, e.g. NIF.1 and indicate the NIF type. There are 3 possible types: 1) a direct connection by means of a serial port;
2) a network connection by means of an arcnet adapter; 3) a modem connection by means of a serial port. In this example, we opt for a direct connection by means of a serial port. Subsequently select the <Add

NIF> button.

3.3.12 Step 12: Settings of a NIF



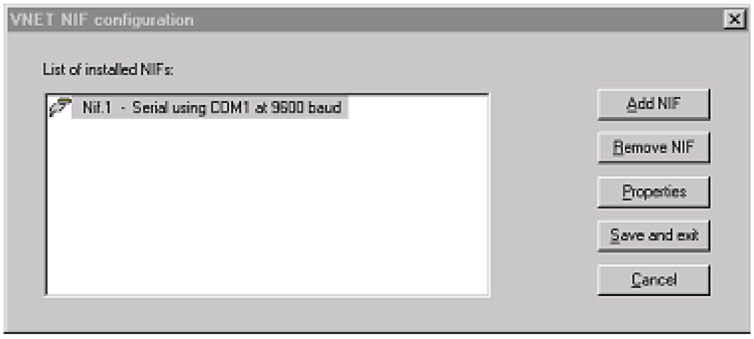
Description: Serial using COM1 at 9600 baud

NIF settings
Portname: COM1
Node number: 160
Serialize: ☐

Port settings
Baudrate: 9600
Tries: 2
Modem delay: 350

Select the correct serial port (COM1 or COM2), set the baud rate of the connection (normally 9600 baud). For the moment the other parameters can keep their standard value. Then click with the left mouse button on the <OK> button. If you haven't entered a valid description in the Description field, the program will choose a description that represents the chosen properties.

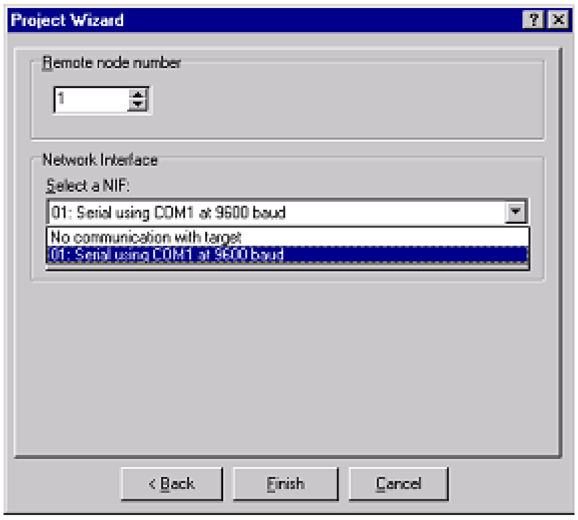
3.3.13 Step 13: The new NIF has been created



List of installed NIFs:
Nif.1 - Serial using COM1 at 9600 baud

Move the mouse towards the <Save and exit> button and click on it with the left mouse button. The new NIF has been created and we can return to EWB32.
If you use the <Cancel> button, all the executed changes are lost.

3.3.14 Step 14: Select the new NIF



Remote node number: 1

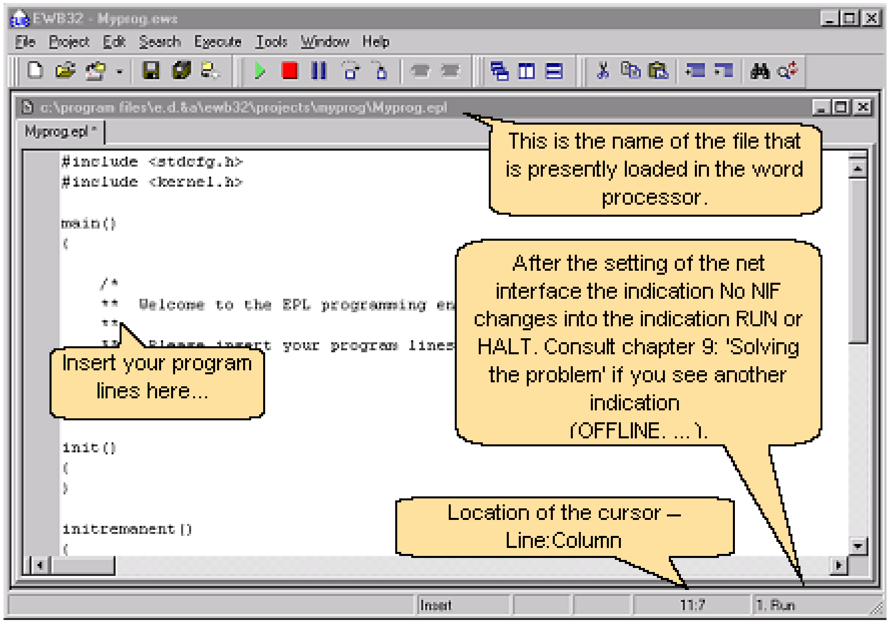
Network Interface

Select a NIF: 01: Serial using COM1 at 9600 baud

No communication with target
01: Serial using COM1 at 9600 baud

Choose this newly created NIF from the list.
We have inserted all the necessary parameters of our new project. If we still would like to change something, we can return to the previous screen by using the <Back> button. If all the settings are correct, you use the <Finish> button to open the new project.

3.3.15 Step 15: Insert the program lines


EWB32 - Myprog.ews
File Project Edit Search Execute Tools Window Help
c:\program files\e.d.&a\ewb32\projects\myprog\Myprog.epl
Myprog.epl *
This is the name of the file that is presently loaded in the word processor.
After the setting of the net interface the indication No NIF changes into the indication RUN or HALT. Consult chapter 9: 'Solving the problem' if you see another indication (OFFLINE, ...).
Insert your program lines here...
Location of the cursor – Line:Column
Insert
11:7
1. Run

```
#include <stdcfg.h>
#include <kernel.h>

main()
{

    /*
    **  Welcome to the EPL programming en
    **
    ** Please insert your program lines


init()
{
}

initremanent()
{
```

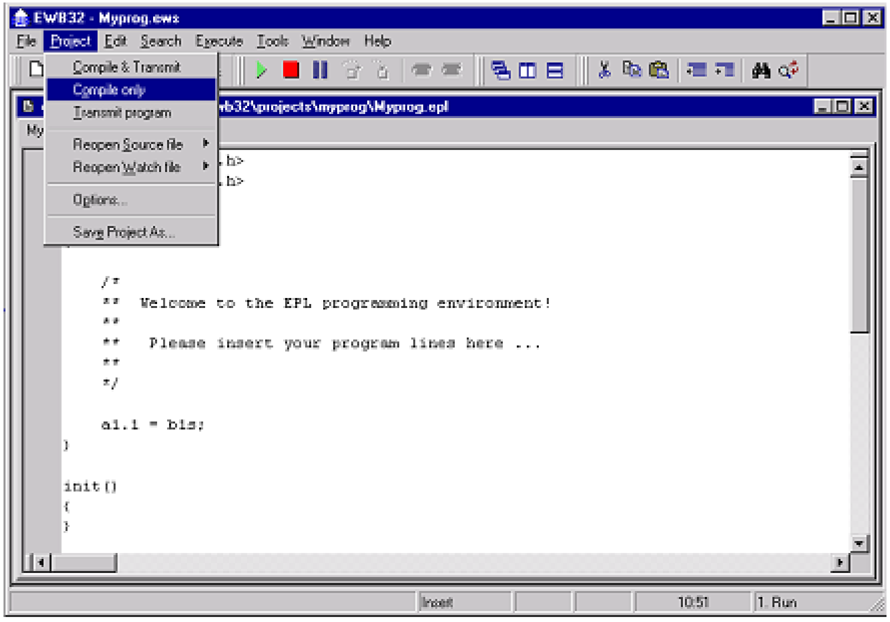
After the project settings are completed, the project management loads an elementary EPL program in the word processor. We will use this basic program as a foundation for our application. Insert the following program line into the program block (function):

```
a1.1=b1s;
```

Use the cursor keys and the mouse for moving the cursor. Make sure that the program line is being closed

off by the symbol ;.
For the use of the word processor we refer to The source code word processor 24

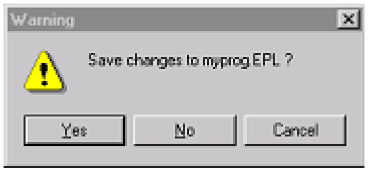
3.3.16 Step 16: Translate the program lines



```
.h>
.h>

    /*
    **  Welcome to the EPL programming environment!
    **
    **   Please insert your program lines here ...
    **
    */

    a1.1 = b1s;
}

init()
{
}
```

After you have carefully added the program line, you select the menu task Project | Compile Only. This task will translate the EPL program into feasible instructions for the HPC. At the same time, the programming environment will check the validity of the program. Possible errors will be made knowable in the 'Compile Results Window'

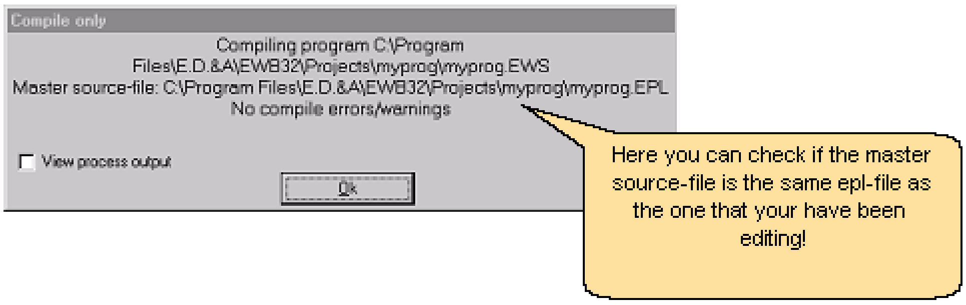
3.3.17 Step 17: Saving the translation



Save changes to myprog.EPL ?

Dependent on the general setting of the program, it is possible that this question appears. The programming environment has noticed that changes have occurred in the file that is loaded in the word processor. Before the compiler is started up, you have to choose whether or not you want to save the changes. Press <YES> to continue.

3.3.18 Step 18: Result of the translator

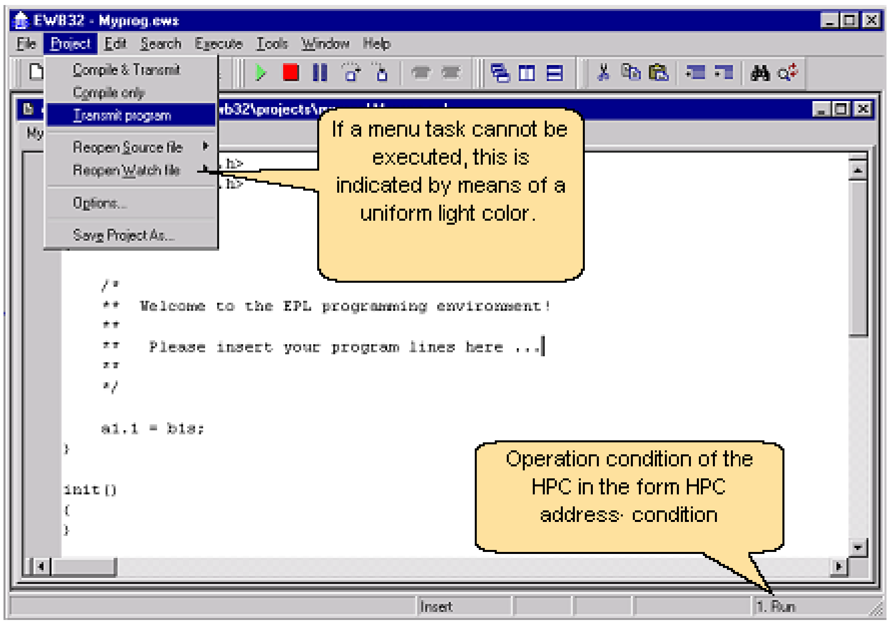

Compile only

Compiling program C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.EWS
Master source-file: C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.EPL
No compile errors/warnings

View process output
Ok
Here you can check if the master source-file is the same epl-file as the one that your have been editing!

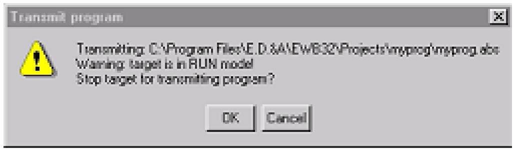
A few moments later the translation will be finished.
If errors have occurred during the translation, the dialog alongside will become visible. Press <OK> to proceed.
If you have made mistakes during the insertion of the program, you have to correct these before you proceed. In this case you have to open the 'Compile Results Window' to check what is going on.
Consult chapter 10: Messages of the compiler 58 for an extensive description of the various notifications of the compiler.

3.3.19 Step 19: Send the translated program to the HPC


```
/*
**  Welcome to the EPL programming environment!
**
**   Please insert your program lines here ...
**
*/

    a1.1 = b1s;
}

init()
{
}
```
EW832 - Myprog.ews
File Project Edit Search Execute Tools Window Help
Compile & Transmit
Compile only
Transmit program
Reopen Source file
Reopen Watch file
Options...
Save Project As...
If a menu task cannot be executed, this is indicated by means of a uniform light color.
Operation condition of the HPC in the form HPC address: condition
Insert
1: Run

After the source program is translated, we have to send it to the HPC before the program can be executed. Select the menu task Project / Transmit Program to send a translated program to the HPC. We can also do the translation and the sending of a program in one step by selecting the menu task Project / Compile & Transmit.

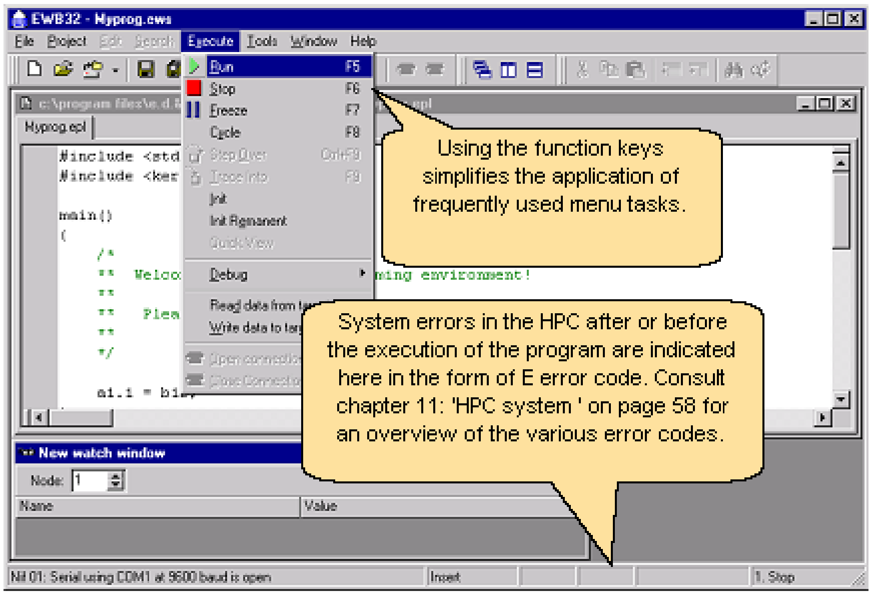
3.3.20 Step 20: Interrupt the program termination


Transmit program

Transmitting: C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.abs
Warning: target is in RUN mode!
Stop target for transmitting program?

If the 'target' HPC is executing a program (RUN mode), it is not always advisable that the program termination is suddenly interrupted to load a new program. This may lead to dangerous situations. If the 'target' HPC is in RUN mode, the programming environment will ask for an explicit confirmation to interrupt the program termination in the 'target' HPC to subsequently send the new program. Press <OK> to proceed.

3.3.21 Step 21: Execution of the loaded program

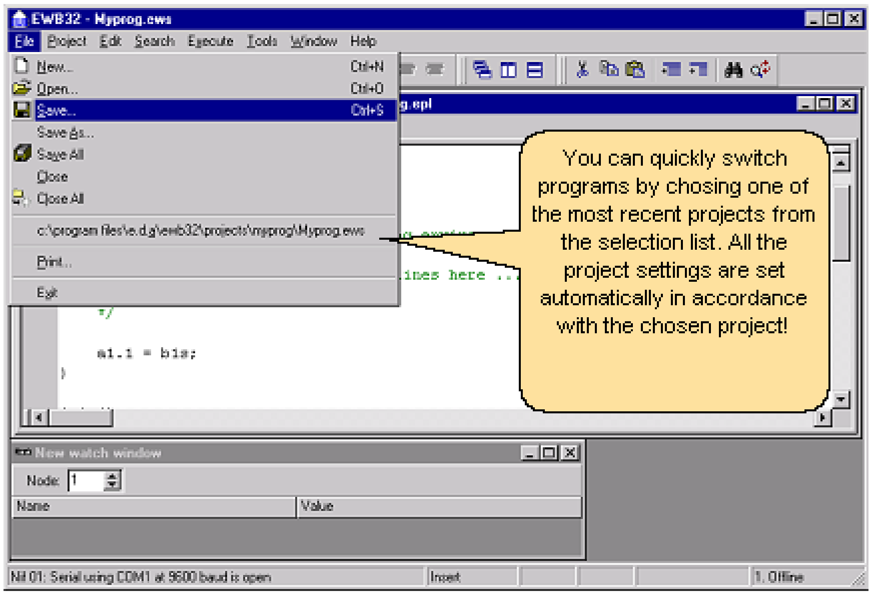


Using the function keys simplifies the application of frequently used menu tasks.

System errors in the HPC after or before the execution of the program are indicated here in the form of E error code. Consult chapter 11: 'HPC system ' on page 58 for an overview of the various error codes.

After the translated program has been sent to the HPC, this is in STOP mode. This means that the program is not being executed. Then choose the menu task Execute / Run to execute the program. By using function key F5 you become the same result.

3.3.22 Step 22: Save all your work



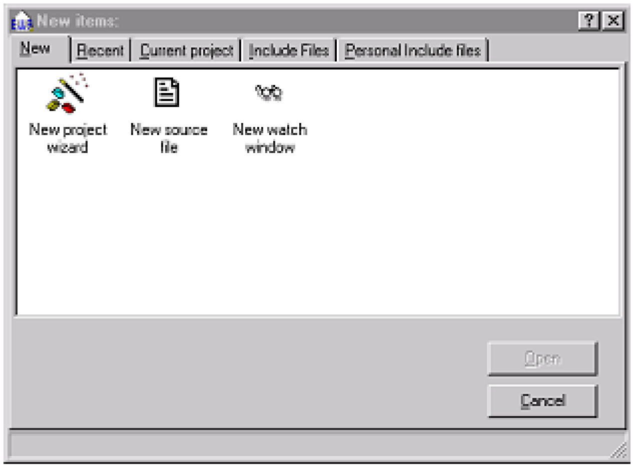
You can quickly switch programs by chosing one of the most recent projects from the selection list. All the project settings are set automatically in accordance with the chosen project!

```
a1.1 = b1s;
```

Nil 01: Serial using COM1 at 9600 baud is open | Insert | 1. Offline

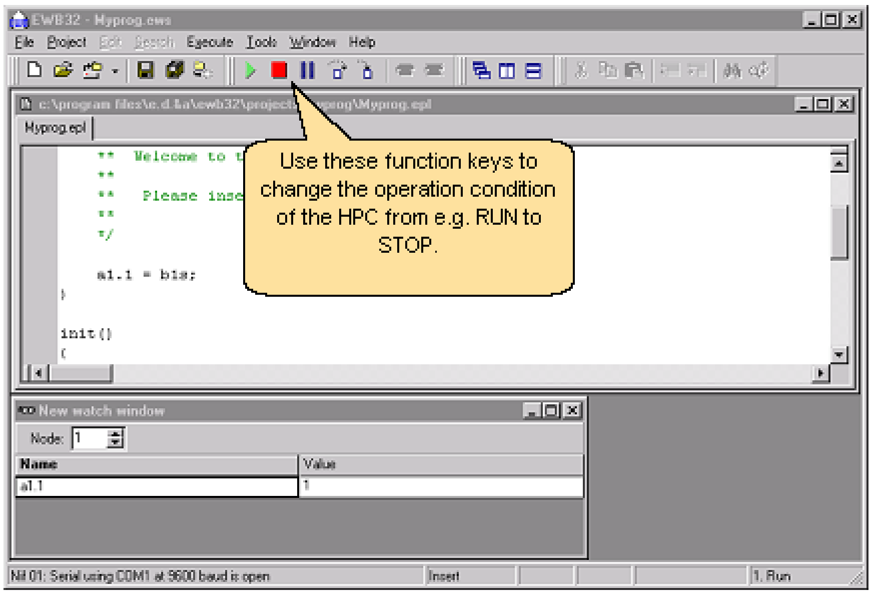
An extra tip: you can always save your work by executing the menu task File / Save or File | Save All.

3.3.23 Step 23: Create a new watch window



To be able to follow the course of the program, there is a possibility to represent the content of program variables in a dynamic way. For the representation of program variables, we use a 'Watch Window'. If you do not have a watch window on your screen, use File | new | New watch window. You yourself can determine how many watch windows you want to put on your screen.

3.3.24 Step 24: Dynamic representation of program variables

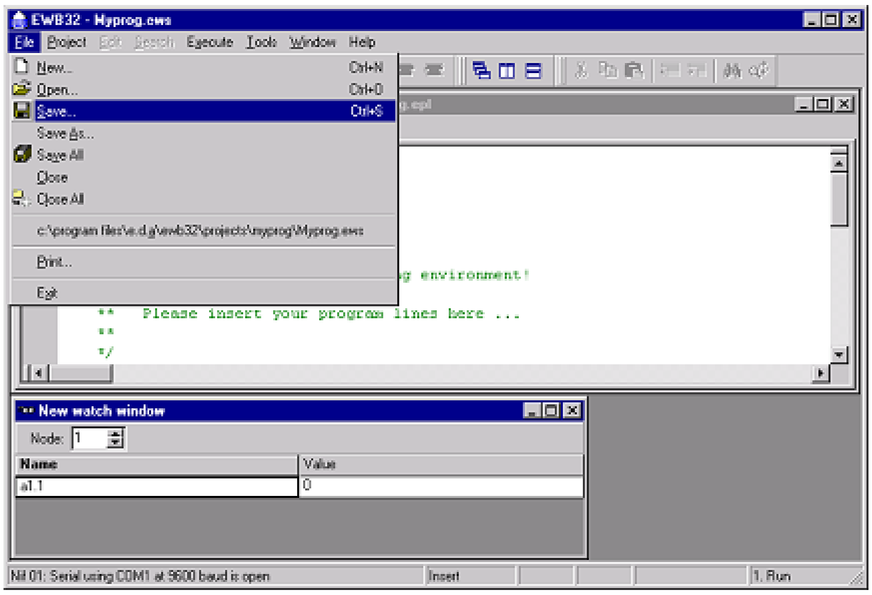

EWB32 - Myprog.ews
File Project Edit Search Execute Tools Window Help
c:\program files\e.d.&a\ewb32\projects\myprog\Myprog.epl
Myprog.epl
Use these function keys to change the operation condition of the HPC from e.g. RUN to STOP.
New watch window
Node: 1
Nil 01: Serial using COM1 at 9600 baud is open
Insert
1. Run

```
    **  Welcome to t
    **
    **   Please inse
    **
    */

    a1.1 = b1s;
}

init()
{
```

Name	Value
a1.1	1

It is enough to accept the name of a program variable in a 'Watch Window' so that these are being dynamically retrieved and represented. Select the window 'New watch window' and insert the text A1.1 (column Name) and close off by pressing the Enter key. You will notice that the value of variable A1.1 is automatically brought up.

3.3.25 Step 25: Save a watch window


EWB32 - Myprog.ews
File Project Edit Search Execute Tools Window Help
New... Ctrl+N
Open... Ctrl+O
Save... Ctrl+S
Save As...
Save All
Close
Close All
c:\program files\e.d.&a\ewb32\projects\myprog\Myprog.ews
Print...
Exit
New watch window
Node: 1
Nil 01: Serial using COM1 at 9600 baud is open
Insert
1. Run

```
g environment!
    **   Please insert your program lines here ...
    **
    */
```

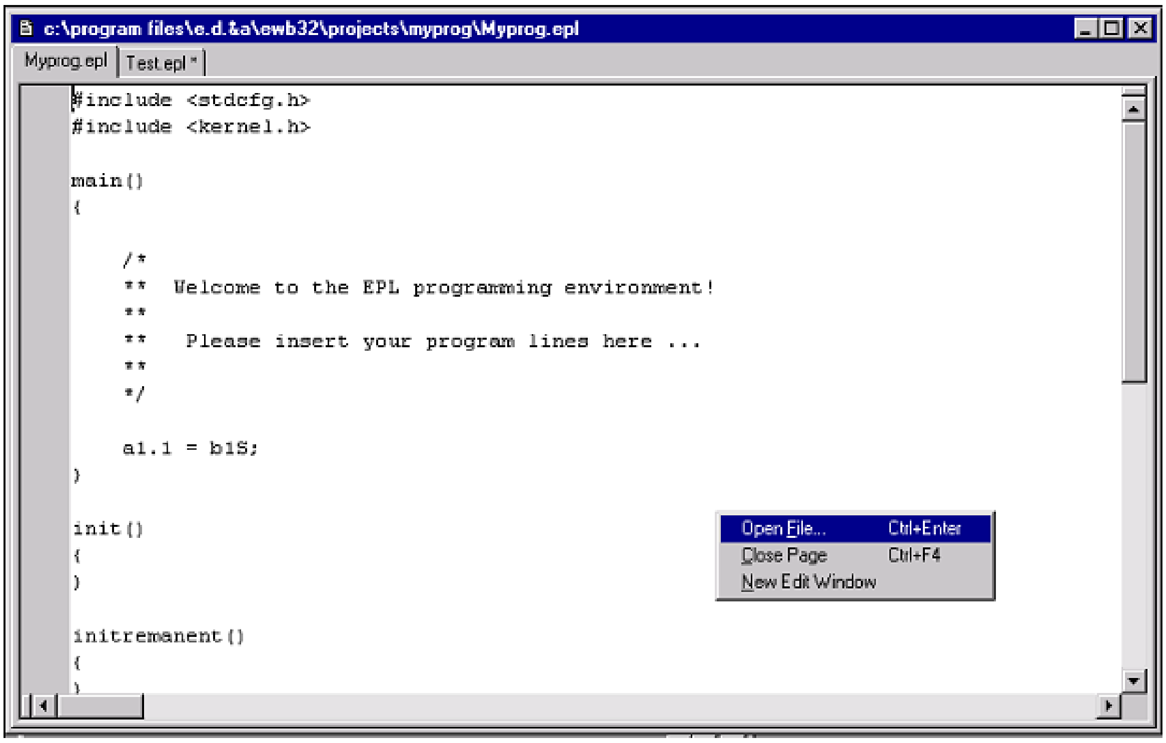
Name	Value
a1.1	0

Just like the files in the word processor, you can also save the content of the 'Watch Windows' on the hard disk. Select the menu task File | Save to save the active 'Watch Window'. The active window has a coloured (usually blue, but it depends on the settings on your system) bar on top. A non-active window has a gray bar. Use the menu task File | Save As to save the 'Watch Window' under another name. The menu tasks in the File menu are dependent on the active window. These are called 'Context Sensitive Menus'.

3.4 And further

Here ends our step by step tour through EWB32.
Now you can make a project on your own, you can insert program lines, translate the program and send it to the HPC.
Dependent on your available time, you can now go into the possibilities EWB32 by reading chapter 5 and following.
However, you can also decide to begin instantly and read the chapters when you want to.
We wish you a lot of programming pleasure!

4 The source code word processor



```
#include <stdcfg.h>
#include <kernel.h>

main()
{

    /*
    **  Welcome to the EPL programming environment!
    **
    **   Please insert your program lines here ...
    **
    */

    a1.1 = b1S;
}

init()
{
}

initremanent()
{
}
```

The source code word processor contains one or more windows. You can open an extra source code word processor window by means of the menu item Window | New edit window or by means of the shortcut menu and the menu item New edit window. The window of the source code word processor always has the name and the complete path of the file you are working at, as a title. Various files can be simultaneously opened . A new file can be opened by means of the menu choice File | new of File | open or Open File in the shortcut menu. You can switch between the opened files by clicking with the left mouse button on the name of the file. At the bottom of the screen the source code word processor indicates the present place of the cursor. The representation is line:column.

4.1 Mark a field

By moving the mouse in the word processor while pressing the left mouse button, you can mark a field. The marking stays possible as long as you keep pressing the left mouse button. The marked field is indicated by means of light characters on a dark background. Once you release the button, you can do special computations in the marked field. If you wish to undo the marking of a field, you have to use the left mouse button once again. But you have to be careful. Operations on marked fields may be huge and you cannot undo them.

4.2 Additional special key functions.

Enter
    Start a new line where the cursor is situated.

Control+Home
    Move the cursor towards the first character of the file.

Control+End
    Move the cursor towards the last character of the file.

Control+Insert
Copy a marked field to the memo block.

Shift+Delete
Remove a marked field after it has been copied to the memo block.

Shift+Insert
Copy the content of the memo block to the location of the cursor or replace a marked field with the content of the memo block.

Shift+F3
Move the cursor to the next location containing an error. This possibility cooperates with the 'compile' task and is only applicable the last time the program has been compiled without changes that have been made after the translation.

Shift+F4
Move the cursor to the previous location containing an error. This possibility cooperates with the 'compile' task and is only applicable the last time the program has been compiled without changes that have been made after the translation.

4.3 Breaking points

In case the option "Create debug code" has been selected in the project options, a breaking point can be placed in the program on the program line. The implementation of the program will be interrupted before the instruction on this line will be executed. At this moment the HPC status will be "trace". (at the right in the status bar)

It is very practical or sometimes even essential to find faults in the program.

In case you want to use breaking points, you need to install the software EWB32 source code debugger (S2-975).

4.3.1 Place breaking points

A breaking point can be placed in 2 ways:

- Click with your mouse on the blue tracing point, to the left of the editor
- Place your cursor with the keyboard on the line and press CTRL+B

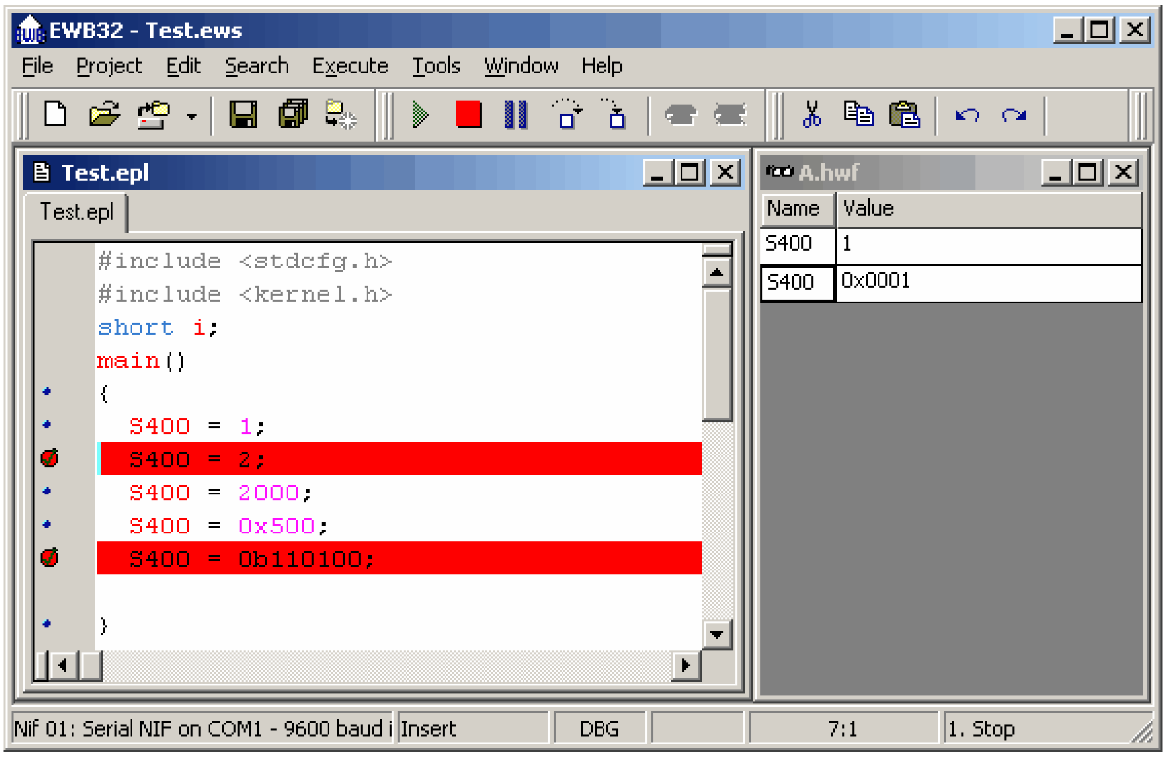
If at an existing breaking point the same action is been executed, the breaking point will be deleted.

Max. 4 active breaking points can be placed. See chapter List of breaking points 27

Example

In this example we give variable S400 each time a different value. Two breaking points are placed, one if

S400 = 1 and the other if S400 =0x0500.

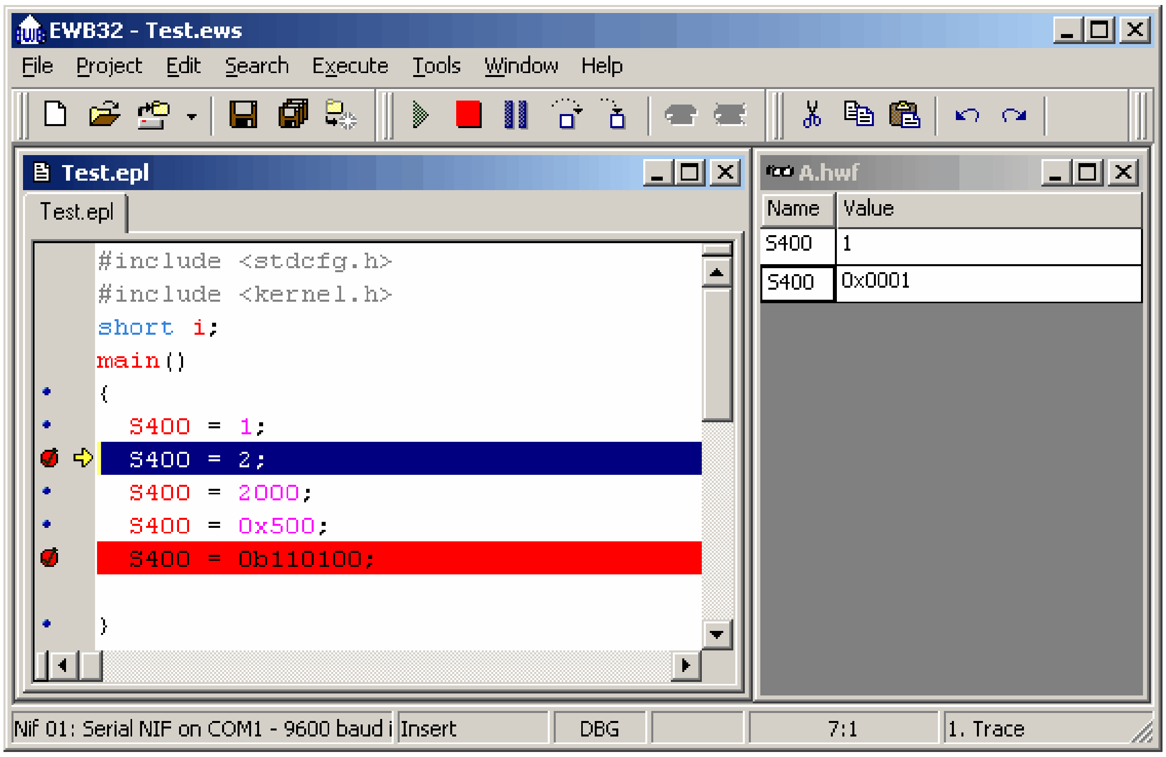


```
#include <stdcfg.h>
#include <kernel.h>
short i;
main()
{
  S400 = 1;
  S400 = 2;
  S400 = 2000;
  S400 = 0x500;
  S400 = 0b110100;

}
```

Name	Value
S400	1
S400	0x0001

Nif 01: Serial NIF on COM1 - 9600 baud i Insert DBG 7:1 1. Stop

If the program starts up (via menu Execute - Run or F5), the program will stop at the first breaking point:



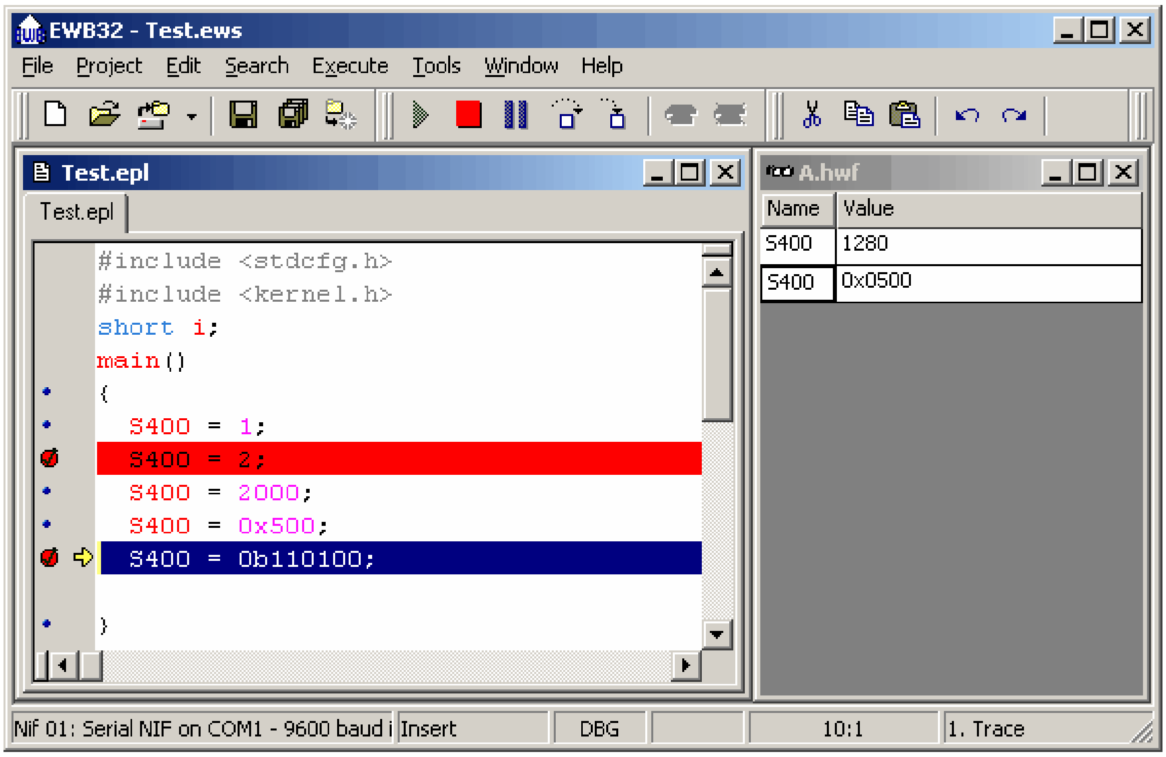
```
#include <stdcfg.h>
#include <kernel.h>
short i;
main()
{
  S400 = 1;
  S400 = 2;
  S400 = 2000;
  S400 = 0x500;
  S400 = 0b110100;

}
```

Name	Value
S400	1
S400	0x0001

Nif 01: Serial NIF on COM1 - 9600 baud i Insert DBG 7:1 1. Trace

As you can see in the watch window S400 equals 1.

If we let the program run (via menu Execute- Run or F5), the program will continue to the next breaking point.


EWB32 - Test.ews
File Project Edit Search Execute Tools Window Help
Test.epl
A.hwf
Nif 01: Serial NIF on COM1 - 9600 baud i
Insert
DBG
10:1
1. Trace

```
#include <stdcfg.h>
#include <kernel.h>
short i;
main()
{
  S400 = 1;
  S400 = 2;
  S400 = 2000;
  S400 = 0x500;
  S400 = 0b110100;

}
```

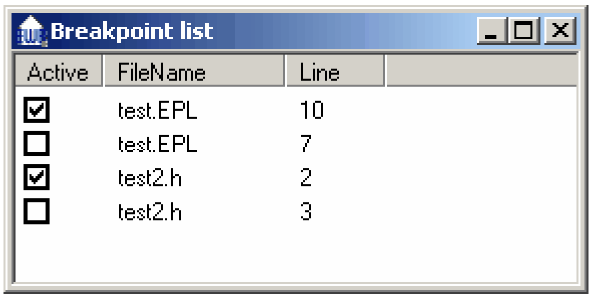
Name	Value
S400	1280
S400	0x0500

Now S400 = 1280 or 0x0500. The program will stop at the 2 breaking points.

4.3.2 List of possible breaking points

There is no restriction on the number of breaking points in a program, but max 4 of the placed breaking points can be active at the same moment.

The breaking points can be managed in the editor, but if there are breaking points in different files, the "Breakpoint list" is practical.

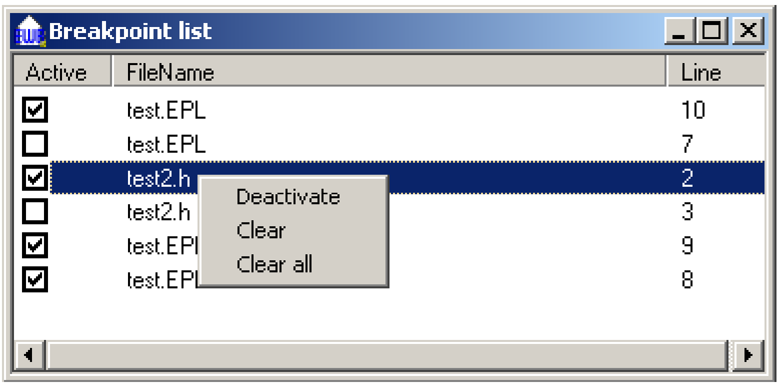


Active	FileName	Line
✓	test.EPL	10
☐	test.EPL	7
✓	test2.h	2
☐	test2.h	3

In this example, there are 4 breaking points: 2 in the file 'test.epl' and 2 in 'test.h'. In the first column the breaking point can be (in)activated. If 4 breaking points are active, it is not possible to select a fifth breaking point. In column 2 and 3 you will find the filenames and line numbers of the breaking points.

You can click at the right to select a breaking point:

- Activate or deactivate: to make a breaking point active or passive
- Clear: to delete a breaking point
- Clear all: to delete all the breaking points



Active	FileName	Line
✓	test.EPL	10
☐	test.EPL	7
✓	test2.h	2
☐	test2.h	3
✓	test.EPl	9
✓	test.EPl	8

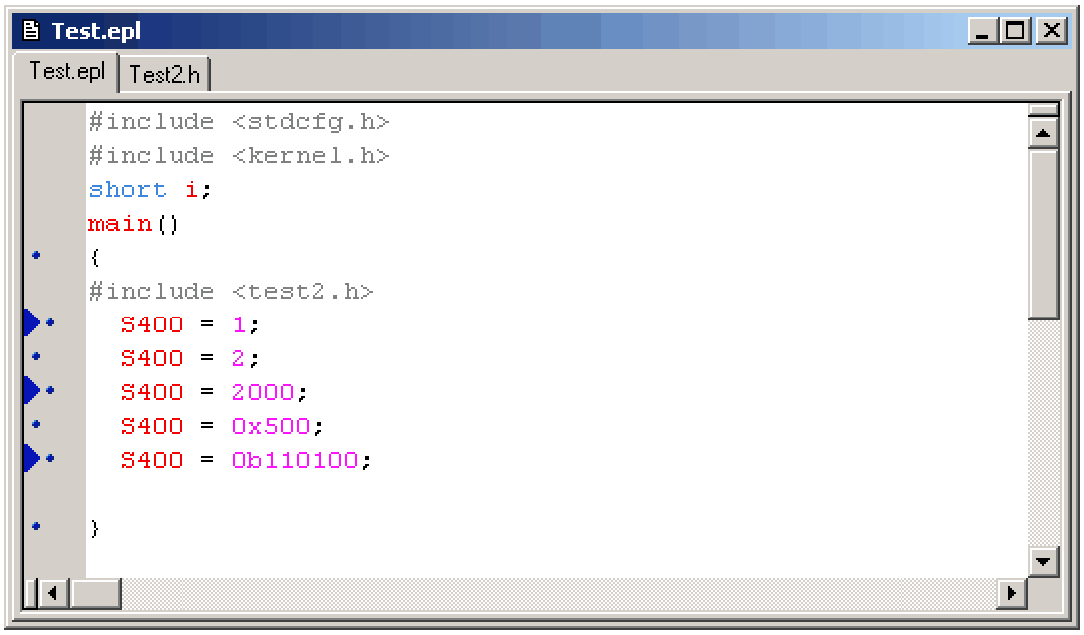
4.4 Bookmarks

When you are programming, you probable have to use the different functions. The use of bookmarks simplifies this.

Place a bookmark
A bookmark can be placed in two steps:

- Place your cursor at the place where the bookmark has to be drawn up
- Press CTRL + F2

A blue triangle will appear on the left next to the line.



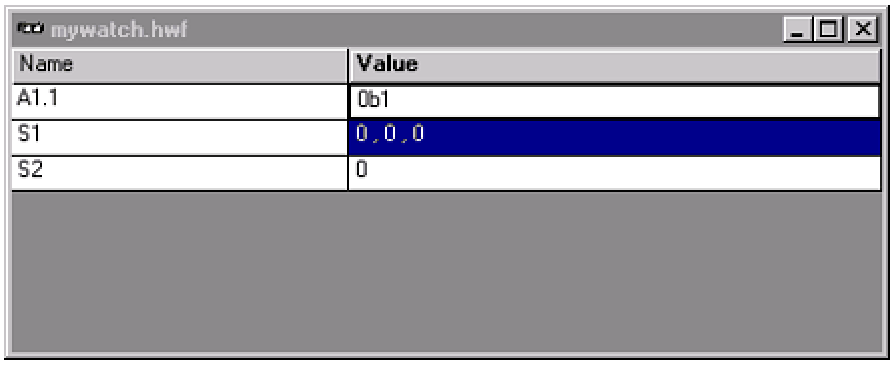
```
#include <stdcfg.h>
#include <kernel.h>
short i;
main()
{
#include <test2.h>
  S400 = 1;
  S400 = 2;
  S400 = 2000;
  S400 = 0x500;
  S400 = 0b110100;

}
```

Jump to a bookmark
In case you press F2, the cursor will go to the first bookmark below the cursor. In case you press SHIFT+ F2, the cursor will go to the first bookmark above the bookmark.

If there are no more bookmarks above or below the cursor, the search restarts at the beginning or at the end of the file.

5 The test mode

5.1 In general


mywatch.hwf

Name	Value
A1.1	0b1
S1	0 , 0 , 0
S2	0

The general purpose of the test mode is representing the content of variables in the HPC. The actual state of a variable is represented in a window, here in 'mywatch'. In the right bottom corner of the screen the state of the HPC is indicated. If the communication between the HPC and the test mode is interrupted, this is indicated by means of the 'offline' or 'Unconnected' message. An interrupted connection between PC and HPC or an HPC that is not switched on, are the most frequent causes. 'RUN' indicates that the HPC is executing a program. 'STOP' indicates that the HPC has interrupted the execution of its cyclic program and its 'event' functions. Also all the digital outputs are switched off and the analogue outputs are switched to 0. 'FREEZE' indicates that the state of the HPC is frozen. The HPC has interrupted the execution of its cyclic program and its 'event' functions. The digital and analogue outputs, however, are still guided, in accordance with the image in the data memory of the HPC.

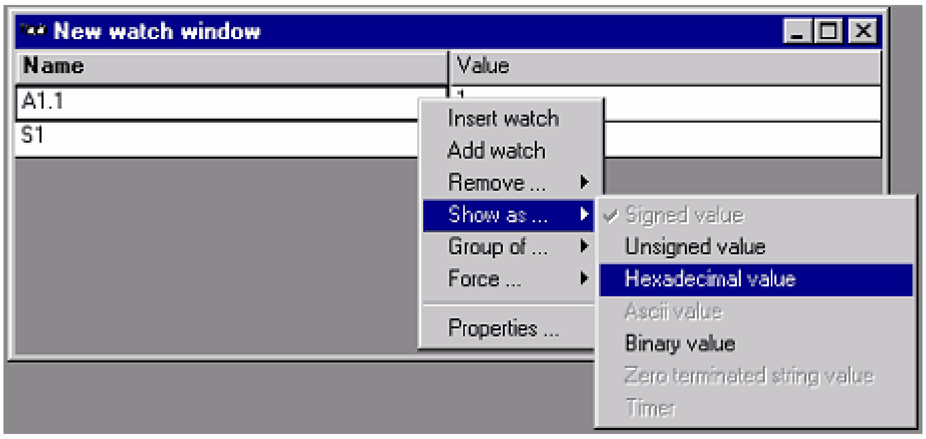
5.2 Representations of variables

5.2.1 Standard representation

The notation, which is used to show a variable on the screen, is identical to the one that is used in the HPC program. The notation for representing a physical output 2 on card 7 is a7.2. This counts for all the physical inputs and outputs and all the word variables. The representation of the value of a variable on the screen is intrinsic to the width of the word of a variable. The variable of the bool type will be represented with a 0 or 1 value. A variable of the char type is represented in decimal notation without a sign (0-255). A short word and a long word are represented in decimal notation with sign.

5.2.2 Free representation

If the standard representation of a variable does not comply with the wishes of the user, it can be changed. The user can adapt the representation of a variable in the windows of the test mode to his specific wishes.



Select the field of the variable of which you wish to change the representation. Click with the right mouse button in this field and the shortcut menu of the test mode appears. Select the menu item Show as…, here you can choose from a number of representation possibilities.

Ascii value
This format converter shows a variable in the so-called character representation. The variable is represented in decimal notation without sign (0-255) and hexadecimal notation (0x00-0xff). Also the corresponding ASCII sign is shown. Notice that this is the standard representation for the char. variable.

Signed value
This format converter shows a variable in the so-called decimal notation. The variable is represented in decimal notation with sign. A variable of the 'bool' type, however, is always represented as a decimal number without a sign.

Unsigned value
This format converter shows a variable in the so-called decimal representation. The variable is represented in decimal notation without a sign.

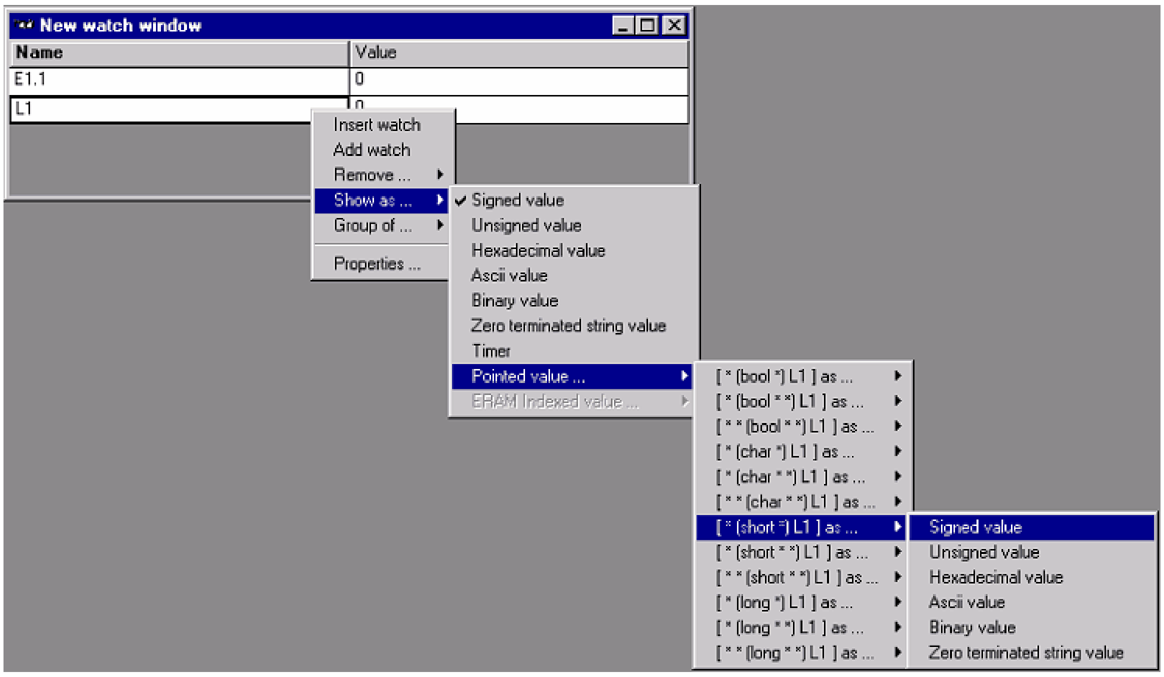
Hexadecimal value
This format converter shows a variable in the so-called hexadecimal representation. The variable is represented as a hexadecimal number (0x..).

Binary value
This format converter shows a variable in the binary representation. The variable is represented as a binary number. Every bit (binary digit) is represented as a 0 or a 1. The number of bits shown on the screen depends on the width of the variable. For a variable of the 'bool' type, it is 1 bit. For a variable of the 'char' type, there are 8 bits. For a variable of the 'short' type there are 16 bits and for a variable of the 'long' type there are 32 bits.

Timer
This format converter shows a variable in the 'timer' representation. This option is only applicable to variables of the 'long' type. In this representation is shown whether the timer that corresponds with the variable, is or is not activated, and how much time has gone by since the timer has started.

Zero terminated string value
This format converter shows a variable in the so-called text representation. The content of the indicated variable and the next one is represented as a text. A 'zero' sign as content of a variable indicates the end of the text.

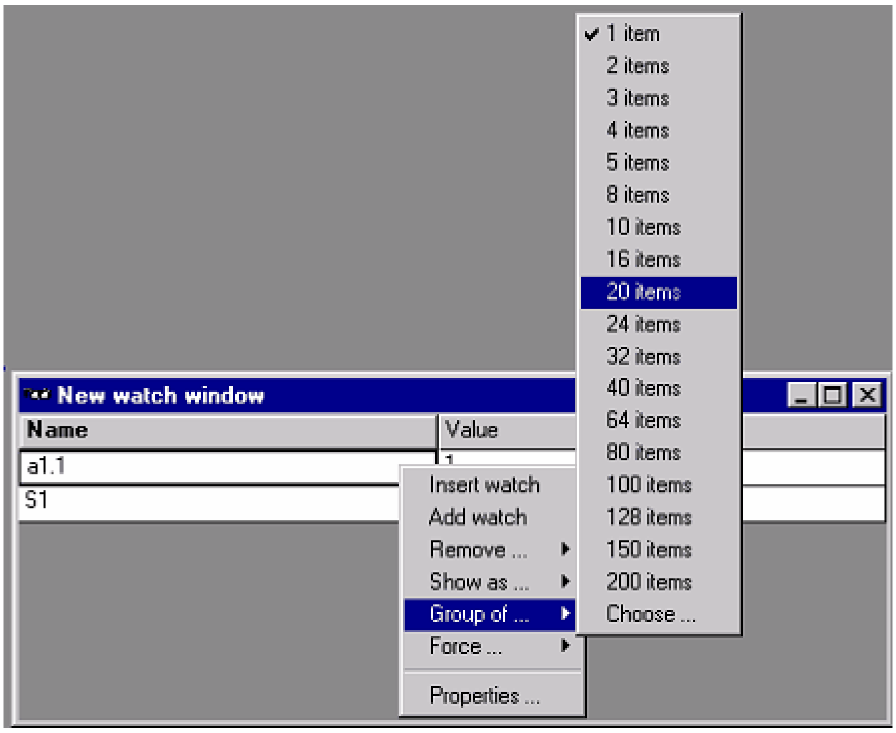
Pointed value
This format converter is only for variables of the long type. You need to choose the type of pointer and the correct level of indirection. For more information about pointers, see the EPL reference manual.



ERAM indexed value

This converter only applies to variables of the long type. The value of the long variable is considered to be as an offset in the ERAM memory. You have to choose between the types of variables and the preferred reproduction of the variable. The use of the ERAM memory is described in the library function ERAM.H.

5.2.3 Group of variables

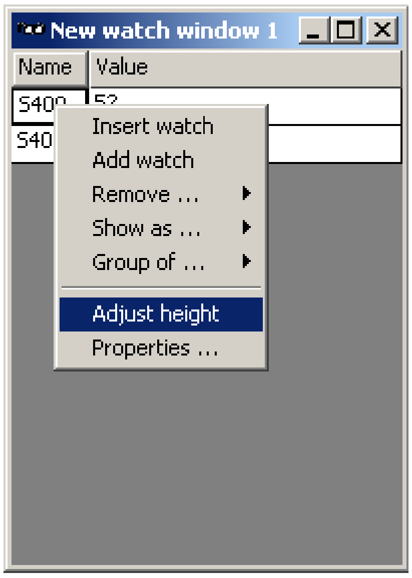


By means of this menu choice you can set to show a group of successive variables of the same type, starting with the variable that was specified. By means of the 'choose' menu item you can administer a number between 1 and 200. The number of variables that is shown in this group corresponds with that number. If you administer more variables than possible (e.g. A1.1 -> 10 pieces) then the watch window will show the following value in the Value column "Variable(s) not configured". This means that you will have to lower down the amount of variables for the selected group.

5.3 Dynamic representation of variables

In the test mode window the content of the variables is represented dynamically in the required format. The content of all the variables in these windows is adapted constantly to the actual content in the memory of the HPC. If << Currently unknown >> is shown, it is impossible at that time to represent the content of the variable. An interrupted communication between PC and HPC can cause this. Adding a variable to the list of desired variables is very easy. Position the mouse on the gray background of the test mode window and press the right mouse button. You see a choice menu where you choose Add watch or Insert watch. A new line is added or inserted. Then you choose a variable from the list or you type the variable you wish to represent. You can only put one variable on a line. If you wish to change a represented variable or its format, you have to select it by means of the mouse or the keyboard and press the right mouse button. The input of the change of an variable that already exists becomes valid from the moment the cursor leaves the line. This happens by pressing enter or one of the cursor control keys. But it is also possible to press the left mouse button outside the input field concerned. Removing a variable from the list happens in a similar way. Select the line concerned, press the right mouse button and select Remove watch. To remove all the watch items simultaneously you can use Remove all watch. Be careful. These operations are irrevocable.

5.4 Adjust cell dimensions



The optimal height for the reproduction of a variable can automatically be adjusted. You have to click with the right mouse button in the screen and then select "Adjust height".The height of the selected cell is adjusted so that all the information is visible in both columns.

5.5 Change the content of a variable

5.5.1 Valid content for a variable.

The user can choose from different forms of constant values for changing the content of a variable.

Decimal constant.
A decimal constant consists of the numbers 0 up to and including 9. Both positive and negative values are allowed. The use of spaces in the number is not allowed. The width of the constant value may not exceed the width of the variable. If the width of the constant is smaller than the width of the variable it will automatically be broadened to the full width of the variable. With a variable with sign this will happen by means of a so-called sign extend, with a variable without sign this is obtained by placing additional zeros.

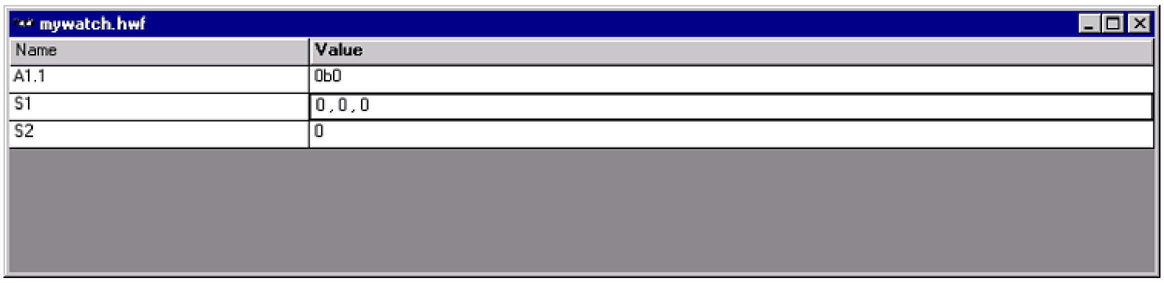
Hexadecimal constant.
A hexadecimal constant consists of the numbers 0 up to and including 9 and the letters A up to and including F. This constant has to be immediately preceded by the indication 0x. The use of spaces in the constant is not allowed. The width of the constant value may not exceed the width of the variable. If the width of the constant is smaller than the width of the variable, it will automatically be broadened to the full width of the variable. With a variable with sign this will happen by means of a so-called sign extend, with a variable without sign this is obtained by placing additional zeros.

Binary constant.
A binary constant consists of the numbers 0 and 1. This constant has to be immediately preceded by the indication 0b. The use of spaces in the constant is not allowed. The width of the constant value may not exceed the width of the variable. If the width of the constant is smaller than the width of the variable, it will be automatically broadened to the full width of the variable. With a variable with sign this will happen by means of a so-called sign extend, with a variable without sign this is obtained by placing additional zeros.

Character constant.
A character constant consists of one up to four printable ASCII signs, preceded and closed off by the sign '. A non-printable sign can be inserted by using a so-called escape sequence. This is the \ sign followed by a single letter or by a number. For the use of the escape sequences we refer to the chapter concerned.

Textual constant or array.
A textual constant consists of an array of zeros or more ASCII signs preceded and closed off by the sign ". A non-printable sign can be inserted by using a so-called escape sequence. This is the \ sign followed by a single letter or by a number. For the use of the escape sequences we refer to the chapter concerned. The ASCII sign NUL (0x00) automatically closes off this array of signs.

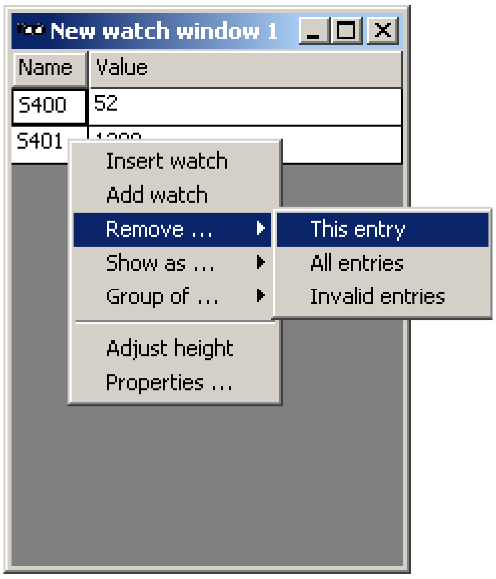
5.5.2 Changing a variable



Name	Value
A1.1	0b0
S1	0,0,0
S2	0

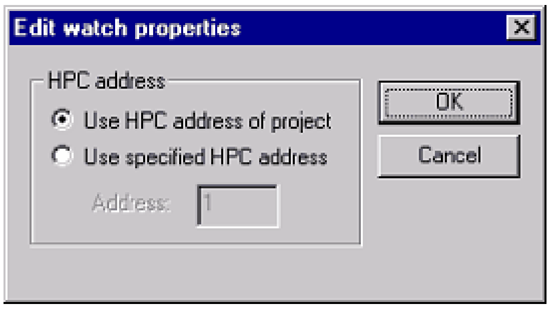
To change the content of a variable that has been dynamically shown , you must proceed as follows. By using the mouse or the keyboard you select the 'Value' column of the variable concerned and then you insert the new content for this variable. You can break off the adjustment of a variable at any time by removing the insertion. Use one of the methods in chapter Representations of variables 30 for inserting the new content of the variable. To change a group of variables, you have to separate the different values by means of commas. If you do not wish to give a certain item in the group a new value, you again give that position a comma.

5.6 Remove variables



With the Remove-item you can delete items from the watch window. If you select “Current Entry” the selected item will be deleted, “All entries” will permanently delete all the items in the window and “Invalid entries” will remove all the items from the active window that have a value of << Invalid …. >>.

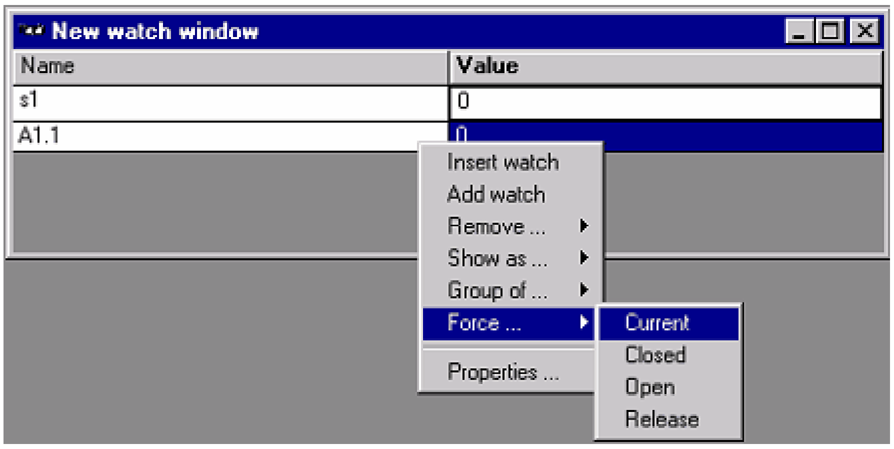
5.7 Change the HPC adres of the testmode



If you have an HPC network, you can view the data of another than the active HPC. To do this, you must enter the HPC address of the HPC in question in the watch window. Click with the right mouse button on the window and select "properties". If the selection "Use HPC address of project" is selected, the watch window will read data from the HPC that is chosen in the project properties. If "Use specified HPC address" is selected, the watch window will show data of the HPC with the specified address.

If you choose an HPC that contains another compiled EPL-program (not the active one), the data that is displayed can be completely wrong, because the place of variables in memory after each compilation will differ. In this case you can not rely on the values that are showed. Changing the HPC address in the watch window is only useful if every HPC in the network has the same compiled EPL-program loaded.

5.8 Force inputs/outputs



If you add an input or output (analogue or digital) in the watch window, you can use the right mouse button to select the Force-menu for this item. You can force (freeze) the selected item in a certain state. The result will be that the EPL-program will not follow the real read value for an input and it will not write the calculated value of the output to the item.

Current
It will Force the value of the item to the momentary value.

Closed
It will Force the value of the item to the closed value.

Open
It will Force the value of the item to the open value.

Release
This will release the Force-command for the item.

6 The draw menus

A draw menu is a menu that is normally hidden for the user. Its presence is only shown by a symbolic name in the menu bar on top of the screen where the menu can appear. Selecting a menu happens by simultaneously pressing the ALT key and a selection sign or by going with the mouse to the symbolic name of the menu and using the left mouse button. At that moment the complete content of the menu will appear on the screen. It seems like the menu is being drawn out. In many cases you can open another draw menu by using the arrow keys or by clicking on another symbolic name. Selecting an option in the draw menu happens by using the selection sign on the keyboard or by moving the mouse towards the option and pressing the left mouse button. You can always close a draw menu by using the ESCAPE key or by moving the mouse outside the draw menu and pressing the left mouse button. It is possible that certain options in the draw menu cannot be selected. This is made visible by a difference in color or intensity. Whether an option can or cannot be selected, is partly determined by the element that is active at the time. Various options also have the possibility to be started up by one single touch. If an option is equipped with this possibility, than it is mentioned after the option. For selecting such options it is not necessary to open a draw menu. Using the above mentioned shortcut key will do.

6.1 The FILE menu

In this menu the user finds all the tasks concerning the HPC program files.

6.1.1 FILE - NEW

The program opens a general dialog screen to open files. This dialog screen shows the tab 'New'. Dependent on your selection, you can 1) create a new project by means of the New Project wizard, 2) create a new source file (.EPL) or 3) open a new test mode window (.HWF).

6.1.2 FILE - OPEN

The functioning of this task is dependent on the window that is active at the time that the menu task is being selected. If you have not opened a project yet, it offers the possibility to open a project (*.ews) In the other case it makes it possible for you to open a source code file (*.epl, *.h) or a test mode file (*.hwf). The program again opens the general dialog window to open files, the active tab, however, is now called "Current project". You can change between maps and filter on other extensions. Select the file of your choice and click with the left mouse button on the <open> button. If you open an existing project while there already is a project opened, EWB32 will close the active project and open the new one. If you have made any changes, the program will ask you to save these first. If the option "AutoSave Editor Files" is switched on, the changes will be saved automatically. You can also select other tabs in the dialog window. For explanation about the tab New I refer to the menu item File-New. The tab Recent contains all the files that were opened recently. The tab "Include-files" allows you to quickly open the EPL include-files with the library functions. The tab "Personal include-files" shows you your personal include-files. For setting these maps I refer to the chapter TOOLS – GLOBAL OPTIONS – FILE LOCATIONS 48.

6.1.3 FILE - SAVE

The functioning of this task is dependent on the window that is active at the time the menu task is being selected. If the word processor is active, the file in the word processor is being saved on disk with the chosen name.
If a watch window is active, the content of the active watch window is being saved on disk with the chosen name.

6.1.4 FILE - SAVE AS...

The functioning of this task is dependent on the window that is active at the time this menu task is being selected. If the word processor is active, the file in the word processor is being saved on disk. The user

can, however, change the name of the file before it is being saved.
If a watch window is active, the content of the active watch window is being saved on disk. The user can, however, change the name of the file before it is being saved.

6.1.5 FILE - SAVE ALL

This task allows you to quickly save all the changed files on disk. Both the files in the word processor and the files in the watch windows are being saved.

6.1.6 FILE - CLOSE

The functioning of this task is dependent on the window that is active at the time this menu task is being selected.
If the word processor is active, the active source code file is being closed. If a watch window is active, it is closed.

6.1.7 FILE - CLOSE ALL

This option allows you to quickly close the open project and all the accompanying files. If changes have been made, the program will ask you to save these first. If the option “AutoSave Editor Files” is switched on, the changes will be saved automatically.

6.1.8 FILE - <RECENTLY OPENED FILES>

The EWB32 shows you up to a maximum of 10 project files in the order they were opened. The most recently opened project file heads the list, the least recently opened file is situated at the bottom of the list. Select a project file from this list to re-open that project. The various settings that are typical of that project are also automatically set.

6.1.9 FILE - PRINT

Printing source code files on a printer.

6.1.10 FILE - EXIT

With this option the user leaves the HPC programming environment.

6.2 The PROJECT menu

The HPC programming environment is project oriented. This means that the system helps the user to create and manage various projects. A project contains all the specific settings necessary for the realization of automation. The project management helps you to maintain the various versions of one and the same project, to create documentation and also helps with the management of the various files a project can exist of. Before you can develop a program, you first have to select or create a program.

6.2.1 PROJECT - COMPILE & TRANSMIT

Before an EPL program can be executed by the HPC, it first has to be translated in tasks executable by the HPC. During the translation the whole program is being checked for possible mistakes. These include errors concerning sentence structure, non-executable tasks, etc. We can distinguish two types of mistakes.

A first group contains all the so-called warnings. These are errors that question certain tasks for the HPC because they may result in possible unexpected results. The translation of the program is not broken off by these warnings. It is perfectly possible to execute such a program but you have to bear in mind that the program execution will not fulfill the expectations for 100%. It is, however, always possible, by introducing small changes in the HPC program, to make these warnings disappear. A second group contains the so-

called fatal errors. These are errors that cannot be ignored by the compiler. The user is obliged to correct the errors before the program can be transferred to the HPC. After the HPC program is translated without any problems, it is transferred to the HPC. The translated program is saved in the HPC in a so-called FLASH-EPROM. This is a memory that is programmable more than once and that keeps its content when there is a loss of voltage. At this time it is sufficient to place the HPC in the so-called 'RUN-MODE' to actually execute the HPC program. If desired, the connection with the HPC can be broken off now if no more additional tests have to be executed.
It is important to know that an HPC program in its translated form cannot be converted into an HPC program in textual form. This means that it is the task of the user to make sure he always saves the original source program, if he wants to make future changes in the program. The HPC program is also saved in its translated form on the hard disk.

6.2.2 PROJECT - COMPILE ONLY

If the user only wishes to translate a certain program, he can do this by selecting this option. After the translation the program will not be transferred to the HPC. This possibility allows you to check the structure of the program for errors without transferring it to the HPC. You can transfer a translated program to the HPC at any time by using the menu task PROJECT – TRANSMIT ONLY.

6.2.3 PROJECT - TRANSMIT ONLY

If a program is already present in its translated form on the hard disk, the user can immediately transfer the program to the HPC by means of this task. It is, however, important to write down that possible changes that were made after the translation of the source program, are not translated and thereby will be executed by the HPC.

6.2.4 REOPEN SOURCE FILE

By using this menu choice, you can quickly reopen one of the most recently opened source code files.

6.2.5 REOPEN WATCH FILE

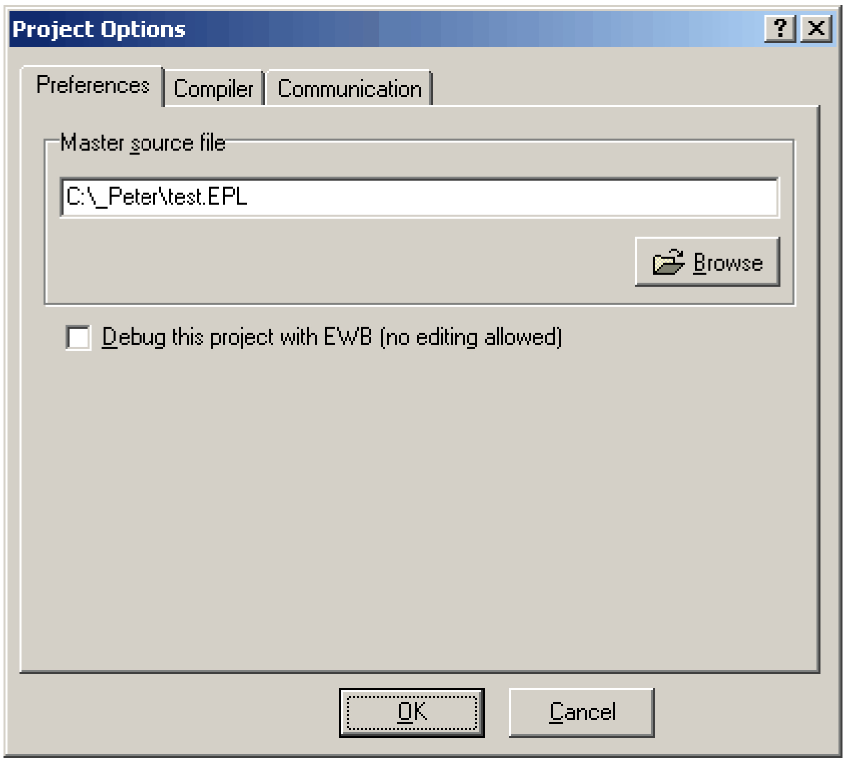
By using this menu choice, you can quickly reopen one of the most recently opened test mode files.

6.2.6 OPTIONS

Under this menu item the user can find all the settings that are specifically related with the active project. The settings were divided over 3 different tabs.

6.2.6.1 Preferences

The master source code file has to be specified here. The master source code file is the file that will be translated first when a project is translated. All the other .EPL or .H files must be called upon from this file.

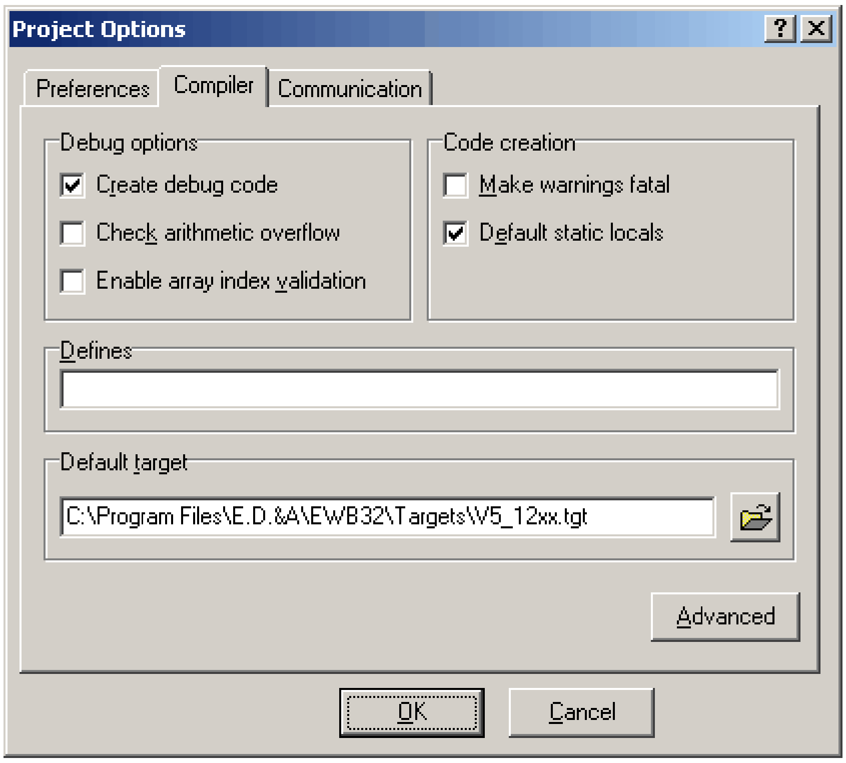


Master source file: C:_Peter\test.EPL

[ ] Debug this project with EWB (no editing allowed)

6.2.6.2 Compiler

Here you can set the settings for the compiler.



Debug options
- [✓] Create debug code
- [ ] Check arithmetic overflow
- [ ] Enable array index validation

Code creation
- [ ] Make warnings fatal
- [✓] Default static locals

Defines

Default target: C:\Program Files\E.D.&A\EWB32\Targets\V5_12xx.tgt

Debug options
These options will only be made visible if you have an EWB32 version with debugger.

Create debug code	when translating the source program, the compiler will add an extra code that allows you to execute the translated program step by step.
Check arithmetic overflow	with this option you can set that the HPC, after every program step, will check whether a certain variable did not have an overflow after an arithmetical limitation.
Enable array index validation	with this option you can set that the HPC, during the index, will check whether somewhere in the program a table index took place outside the area of the table.

Code creation

Make warnings fatal	with this option, you can set that the compiler stops when it has found a warning error in the source code program that has to be translated.
Default static locals	with this you can set that local variables, these are variables with a range within the function where they are made in, do not disappear after you have left the function.

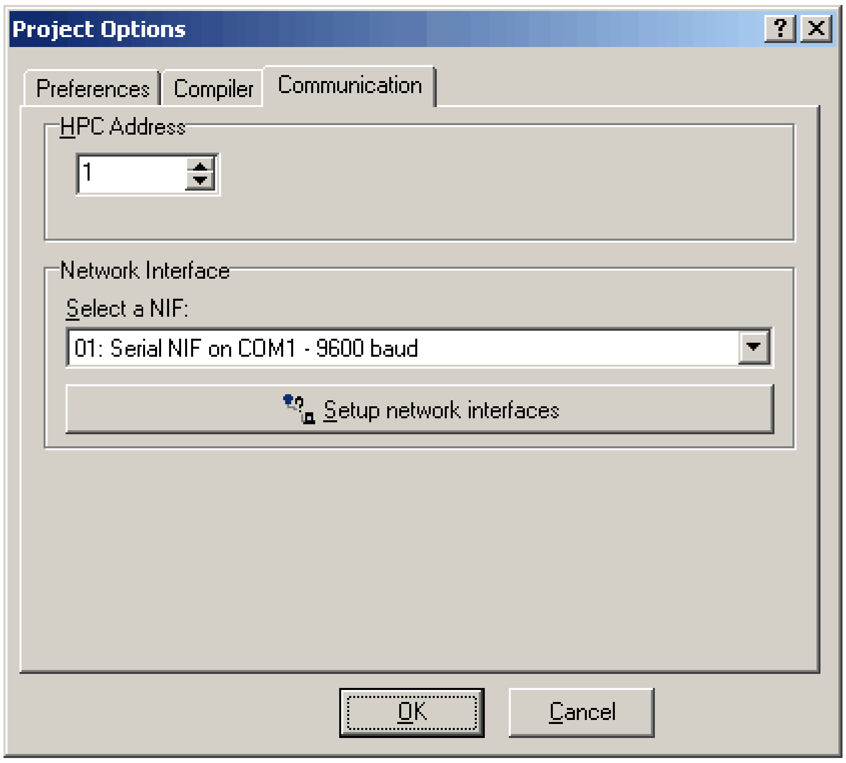
Defines
Here you can declare macro variables that are used by the compiler during the translation of the program and where one can work with in the source code.

Default target
To help the compiler to translate the source program, it is necessary that you indicate for which type of HPC you develop a program. This is described as the 'Target' indication. Click on the map button to set the 'target' HPC. The project management opens a window with the list of installed HPCs for which you can develop a program. The number of different HPC models on the list depends on the number that is installed on your system. The indication of a 'target' or a HPC model corresponds with the order number of the CPU card of the HPC or with the order number of the basic model. Click on the 'target' indication that corresponds with the HPC model for which you wish to develop a program. Then press <OK> to continue.

6.2.6.3 Communication

HPC Address
Specify in the HPC Address field the HPC address that is set on your HPC. For setting the address of your HPC we refer to the Hardware Reference Manual of your HPC. The standard value for this setting is 1.

Network interface
Specify the port that will be used to communicate with the HPC. Choose from the selection list. Click with the left mouse button on the arrow at the back of the input field. If the list only contains the "No communication with target" choice, you can use the <Setup network interface> button to create an interface.
For an example of how to create a NIF, we would like to refer to the step by step chapter of this manual.
Select the "No communication with target" option if you do not wish to make a connection with an HPC.


Project Options

HPC Address: 1

Network Interface

Select a NIF: 01: Serial NIF on COM1 - 9600 baud

6.2.7 PROJECT - SAVE PROJECT AS

If you wish to save your project under a new name, you can use this menu choice. Select the desired map and specify the new name of your project.

6.3 The EDIT menu

In this menu, the user can find all the options that can change texts or that can indicate where the changes have to take place. Many of these options use the memo block. This memo block is a temporary storage for texts. Because you can very easily copy texts from and to this memo block, you can quickly and simply remove large texts, duplicate them and replace them. The use of a memo block also allows you to exchange texts between the various parts of the HPC programming environment. This way you can mark a HPC program example in the help system, copy it to the memo block and insert it in the file in the word processor. The edit menu is connected to the word processor. That is why this menu is only available if the active window is a word processing window.

6.3.1 EDIT - UNDO

By means of this menu choice, you can undo the latest computation in the word processor.

6.3.2 EDIT - REDO

By means of this menu choice, you can repeat the latest computation in the word processor, which was withdrawn by means of undo.

6.3.3 EDIT - CUT

Remove the content of a marked field. The content of the marked field, however, is saved in the memo block.

6.3.4 EDIT - COPY

Copy a marked field to the memo block. The content of the marked field, however, remains unchanged.

6.3.5 EDIT - PASTE

Copy the content of the memo block to the location of the cursor or replace a marked field by the content of the memo block.

6.3.6 EDIT - DELETE

Remove the content of a marked field. The content of the memo block remains unchanged.

6.3.7 EDIT - SELECT ALL

With this you mark the complete content, i.e. all the lines, of the active word processor.

6.3.8 EDIT - INCREASE INDENT

Move a marked field or the current line a few positions to the right.

6.3.9 EDIT - DECREASE INDENT

Move a marked field or the current line a few positions to the left.

6.4 The SEARCH menu

6.4.1 SEARCH - FIND

In this file in the word processor, we look for a specific text.

6.4.2 SEARCH - FIND NEXT

In this file in the word processor, we look for the next appearance of the actual search text. The actual search text can be set by using the menu task EDIT – FIND.

6.4.3 SEARCH - FIND PREVIOUS

In this file in the word processor, we look for the previous appearance of the actual search text. The actual search text can be set by using the menu task EDIT – FIND.

6.4.4 SEARCH - REPLACE

In this file in the word processor, we look for a certain text and, if desired, we can replace it by another one.

6.4.5 SEARCH - FIND IN OPEN FILES

A particular text is searched for in the open files in the word processor.

6.4.6 SEARCH - REPLACE IN OPEN FILES

A particular text is searched for and replaced (if required) in the open files in the word processor.

6.4.7 SEARCH - TOGGLE BOOKMARK

A bookmark is a reference to a certain line in the source code file. Through this option, the user can switch a bookmark on or off on the line where the text indicator is situated. By means of the functions Next and Previous bookmark you can then quickly move the text indicator to respectively the next and previous bookmark in the active file in the word processor.

6.4.8 SEARCH - NEXT BOOKMARK

Move the text indicator to the next bookmark in the active file in the word processor.

6.4.9 SEARCH - PREVIOUS BOOKMARK

Move the text indicator to the previous bookmark in the active file in the word processor.

6.4.10 SEARCH - CLEAR ALL BOOKMARKS

Remove all the bookmarks in the active document.

6.4.11 SEARCH - GO TO LINE NUMBER

Use this menu function in case you wish to move the text indicator to a certain line in the active document. Enter the desired line number.

6.5 The EXECUTE menu

In this menu the user finds all the options that are necessary to guide the execution of a program in the HPC.

6.5.1 EXECUTE - RUN

This option places the HPC in 'run' mode. The HPC executes its program without interruptions.

6.5.2 EXECUTE - STOP

This option places the HPC in 'stop' mode. The HPC interrupts the execution of its cyclic program and its event functions. If the HPC is in 'RUN' mode, the active cycle will be completed first. An active program is never interrupted. All the physical digital outputs are switched off and all the physical analogue outputs are put to 0. The state of the image of the physical outputs in the data memory of the HPC remains unchanged.

6.5.3 EXECUTE - FREEZE

This option places the HPC in 'freeze' mode. The HPC interrupts the execution of its cyclic program and its event functions. If the HPC is in 'RUN' mode, the active cycle will be completed first. An active program is never interrupted. The status of the physical digital and analogue outputs corresponds with the image of the outputs in the data memory of the HPC.

6.5.4 EXECUTE - CYCLE

After the selection of this option, the HPC executes the program only once. The guidance of the physical digital and analogue outputs corresponds with the status of the HPC at the moment the 'cycle' task is given. If the HPC was in 'stop' mode, the physical outputs will not be guided during the execution of the program. If the HPC was in 'FREEZE' or in 'RUN' mode, the physical outputs will take the status of the outputs in the data memory. After ending the cycle, the HPC takes on the status that it was in before it executed the 'cycle' task. Finally, it is allowed, but not very meaningful to give an HPC that is in 'RUN' mode a cycle task.

6.5.5 EXECUTE - STEP OVER

If you have a version of EWB32 with debugger, you can use this menu choice to execute the program line by line in Trace mode. The step over function commands the HPC to execute the source code of the active line and to go to the next line in the active document. If the source code contains a function call, the code of this function is executed continuously.

6.5.6 EXECUTE - TRACE INTO

If you possess a version of EWB32 with debugger, you can use this menu choice to execute the program line by line in Trace mode. The trace into function commands the HPC to execute the source code of the active line and to go to the next line in the active document. Unlike the step over function, the execution of the program stops on the first line of the active function if the source code contains a function call.

6.5.7 EXECUTE - INIT

Selecting this option makes sure that the HPC executes a partly initialization of the system. All the non-volatile data and settings in the HPC receive an initial value. This initialization includes internal parameters in the system, but also variables that will be processed in the HPC program by the user.
The initial value of all non-volatile HPC variables is 0. Variables that are non-volatile, this means variables that keep their value during loss of voltage, are not changed during this initialization. All data and settings in the HPC are being initialized if a HPC is not equipped with a memory data retention or if it does not use it. If the user wishes to give a variable another name during such an initialization of the system, or wishes to execute certain actions, he can do this by using the INIT() function installed in the system. This is an optional function that, in case it is present in the HPC program, will be called upon at the moment that HPC executes a partial (or total) initialization, but after all the non-volatile variables are put to 0. All settings and configurations of the system that are being executed, using the library functions of the HPC, are never non-volatile unless mentioned otherwise. This means that among others the initialization of serial ports and the setting of the 'event' functions always have to take place during every initialization of the system. This may occur by inserting the correct tasks in the INIT() function.

6.5.8 EXECUTE - INIT REMANENT

Selecting this option makes sure that the HPC executes a total initialization of the system. This total initialization makes sure that both the non-volatile and volatile data and settings are being initialized. This initialization includes the internal parameters in the system and the variables that are being executed in the HPC program by the user. The initial value of all HPC variables is 0. If the user wishes to give a non-volatile variable another initial value, he can do this by using the INITREMANENT() function, provided in the system. This is an optional function that, in case it is present in the HPC, will be called upon at the time that the HPC executes a total restart, but after all the non-volatile and volatile variables are put to 0. Then the HPC shall try to call the INIT() function in the HPC program. This is an optional function that, if present in the HPC program, initializes all the volatile data and settings in the HPC. Notice that this function is also called upon by a partial initialization of the HPC. All settings and configurations of the system that are being executed, using the library functions of the HPC, are never non-volatile unless mentioned otherwise. This means that among others the initialization of serial ports and the setting of the 'event' functions always have to take place during every initialization of the system. This may occur by inserting the correct tasks in the INIT() function.

6.5.9 EXECUTE - DEBUG

This menu choice can only be selected if you have a version of EWB32 with debugger. To be able to use these possibilities, the program has to be translated with the "create debug code" option switched on. Attention! This leads to a slower processing of the program because the HPC has to execute extra code. Debugging allows you to purposefully search for errors in the program by executing the source code step by step and by following the course of the program very closely. You can use a breakpoint to interrupt the

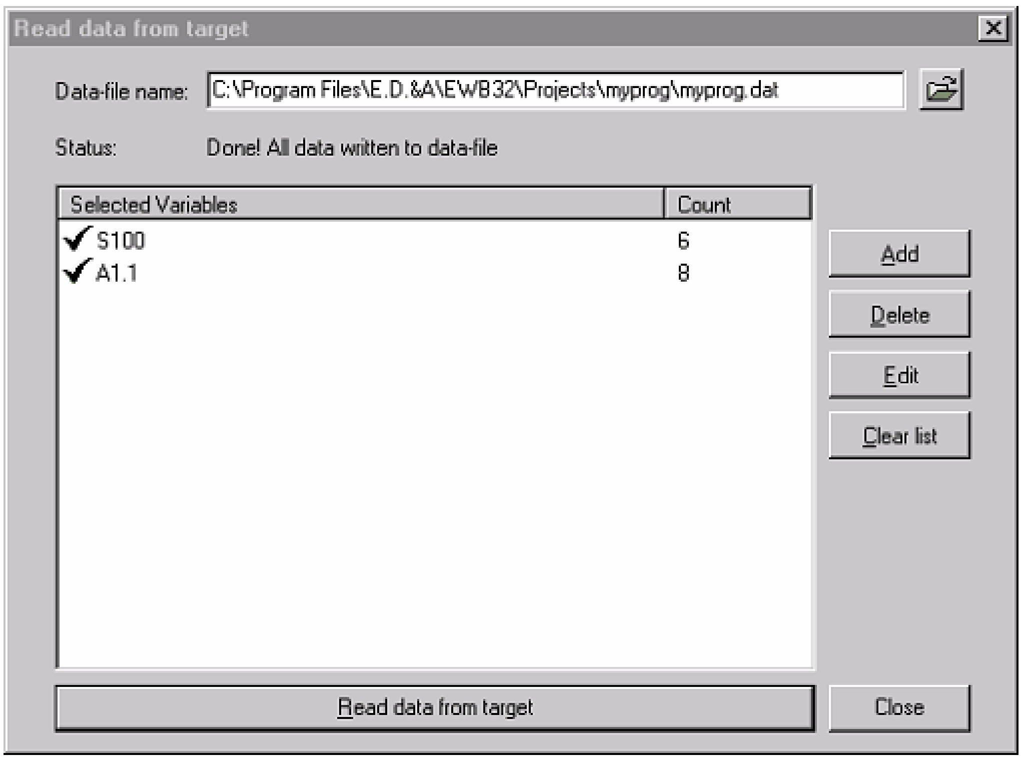
program at that point and you can use the Trace into and Step over functions to get the program under control and continue it step by step.
There is no limit concerning the number of breakpoints, but the maximum number of active breakpoints is 4. Active breakpoints are indicated by a red line, non-active breakpoints by a green line. If you activate a breakpoint on a line that does not contain an executable code, a cross will appear in the ball. In general, lines that have a blue ball in front, contain executable code. When opening a project that contains breakpoints, all breakpoints are put in a non-active mode for security reasons.

EXECUTE - DEBUG - ADD BREAKPOINT
This menu choice allows you to make an indication for the HPC on a certain line in the file in the word processor to interrupt the execution of the program and to wait for further instructions of the user.

EXECUTE - DEBUG - CLEAR ALL BREAKPOINTS
This menu choice removes all the breakpoints in all the files.

EXECUTE - DEBUG - VIEW BREAKPOINTS
Here you get an overview of all the breakpoints. You can determine which breakpoint you want to activate and which you do not want to activate.

6.5.10 EXECUTE - READ DATA FROM TARGET



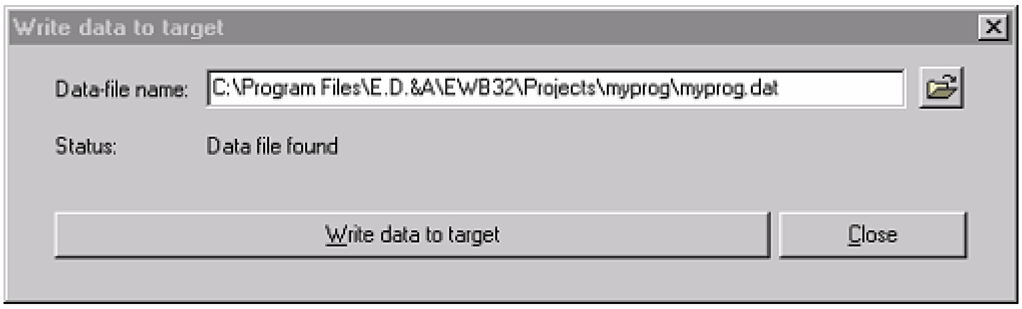
Data-file name: C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.dat

Status: Done! All data written to data-file

Selected Variables	Count
✓ S100	6
✓ A1.1	8

With this menu choice, you can read data (a group of variables) from the HPC and save them in a .DAT file. First indicate the file name of a data file. By means of the <Add> button, you indicate all the variables you wish to read from the HPC. For each variable, you can indicate how many parts you wish to read starting from the memory position of the indicated variable. The <Delete> button can be used to remove a variable from the list, the <Edit> button allows you to change the selected variable or the number you wish to read.
Use the <Read data from target> button to really read the list of variables from the HPC and write them to the data file. It is obvious that the HPC has to be connected with the PC by means of the suited programming cable and that the NIF has to be opened correctly.

6.5.11 EXECUTE - WRITE DATA TO TARGET

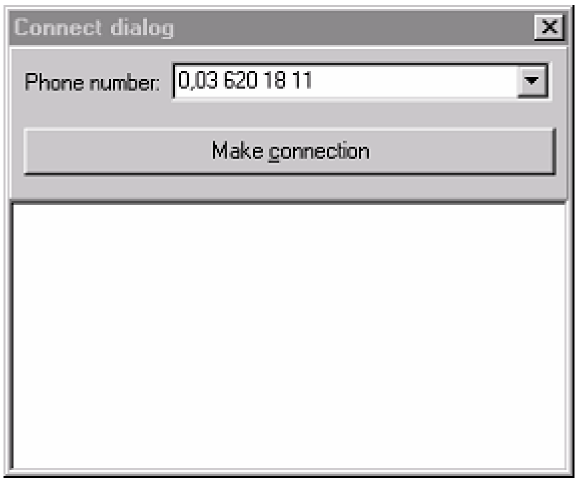


Data-file name: C:\Program Files\E.D.&A\EWB32\Projects\myprog\myprog.dat

Status: Data file found

With this menu choice, you can write data (a group of variables) from a .DAT data file to the HPC. Select an existing data file. Then use the <Write data to target> button to read the list of variables and their values from the data file and write it to the HPC. The .DAT file is an ASCII text file that can be opened by means of a simple word processor. You can change the values of the variables in the file before you send it to the HPC. It is obvious that the HPC has to be connected with the PC by means of the suited programming cable and that the NIF has to be opened correctly.

6.5.12 EXECUTE - OPEN CONNECTION



Phone number: 0,03 620 18 11

With this menu choice, you can write data (a group of variables) from a .DAT data file to the HPC. Select an existing data file. Then use the <Write data to target> button to read the list of variables and their values from the data file and write it to the HPC. The .DAT file is an ASCII text file that can be opened by means of a simple word processor. You can change the values of the variables in the file before you send it to the HPC. It is obvious that the HPC has to be connected with the PC by means of the suited programming cable and that the NIF has to be opened correctly.
The window closes off automatically if the connection with the installation has been realized. Then you can execute all the functions that are possible by means of a direct connection.

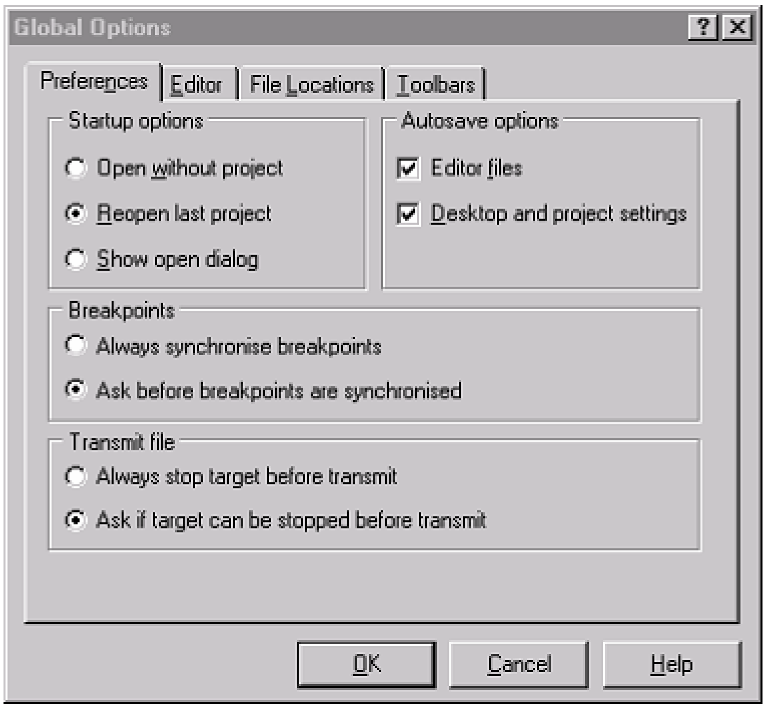
6.5.13 EXECUTE - CLOSE CONNECTION

By means of this menu choice, you break off the connection with the installation at a distance. (Also see EXECUTE - Open connection).

6.6 The TOOLS menu

6.6.1 TOOLS - GLOBAL OPTIONS

TOOLS – GLOBAL OPTIONS – PREFERENCES



Startup options
- ( ) Open without project
- (●) Reopen last project
- ( ) Show open dialog

Autosave options
- [✓] Editor files
- [✓] Desktop and project settings

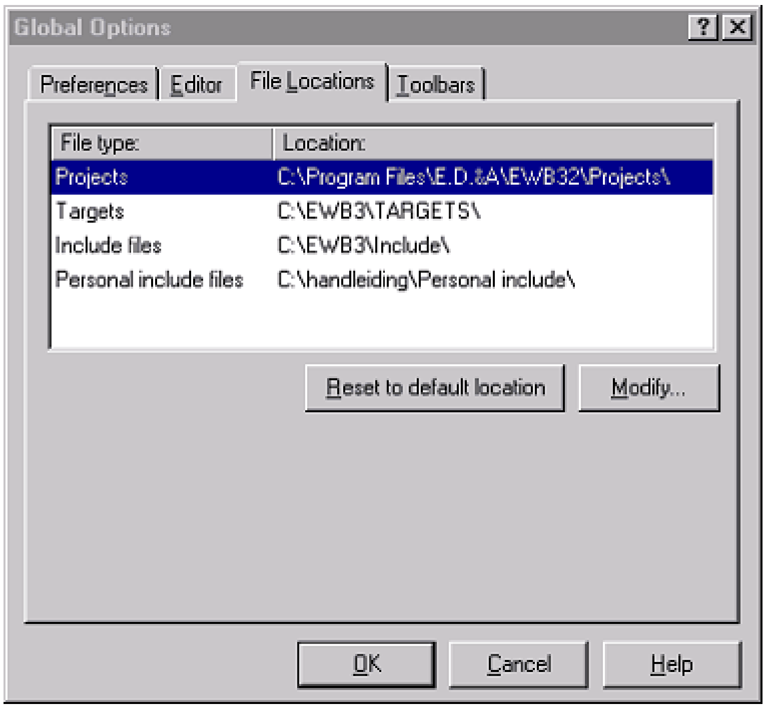
Breakpoints
- ( ) Always synchronise breakpoints
- (●) Ask before breakpoints are synchronised

Transmit file
- ( ) Always stop target before transmit
- (●) Ask if target can be stopped before transmit

This tab contains the settings that allow you to adapt the behavior of EWB32 to your wishes. Set the various options as you desire.

TOOLS – GLOBAL OPTIONS – EDITOR
This tab contains the settings for the word processor. You can set the font and the size of the text as you wish. The list of fonts is limited because the word processor cannot deal with True type fonts.

TOOLS – GLOBAL OPTIONS – FILE LOCATIONS



File type:	Location:
Projects	C:\Program Files\E.D.&A\EWB32\Projects\
Targets	C:\EWB3\TARGETS\
Include files	C:\EWB3\Include\
Personal include files	C:\handleiding\Personal include\

Here you have to indicate the location of the various files. Indicating a wrong pad for the target or the include files, for example, may lead to translation errors during the compilation.

Projects: has to refer to the map where you wish to save your projects.
Targets: has to refer to the map where the target files were installed.
Include files: has to refer to the map where the library files were installed.
Personal include files: has to refer to the map where you wish to keep your personal library files.

TOOLS – GLOBAL OPTIONS – TOOLBARS
Activate the shortcut menus by means of a little cross.

6.6.2 TOOLS - INSTALL NEW TARGET

This is a wizard that guides you during the installation of new target files. Follow the instructions of the wizard to bring the installation to a favorable conclusion.

6.6.3 TOOLS - VIEW INSTALLED TARGETS

This menu choice shows you a list of all the installed target files.

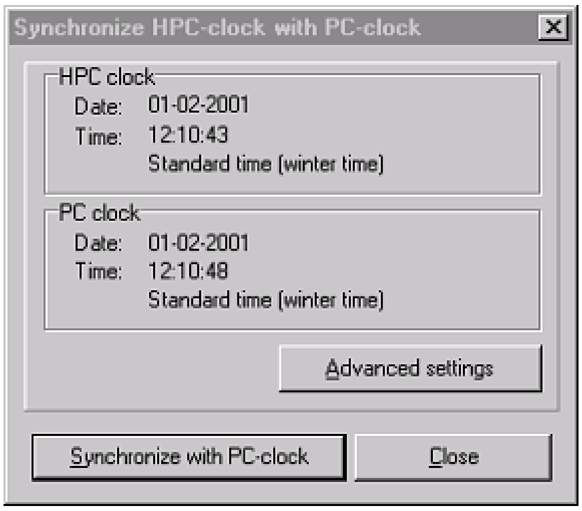
6.6.4 TOOLS - FIND LOCATION

This menu choice allows you, on the basis of the segment/offset address, to find the file and line number of that address. It may occur that your HPC, with an error code E50 or E05, shows the address where the error occurred on its screen. This way you can quickly find which code was executed at the moment of the error.

6.6.5 TOOLS - Synchronise clock

TOOLS – Synchronise clock
The E.D.&A. HPCs can be equipped wit a RTC (Real Time Clock). With this tool, you can synchronise the clock with the system clock from your PC system. Make sure that the HPC is connected to the PC. The clock of the HPC will be read and is displayed in the upper frame. The current time of the PC-clock is displayed in the other frame. Use the button <Synchronize with PC-clock> to write the time of the PC-clock to the clock in the HPC.

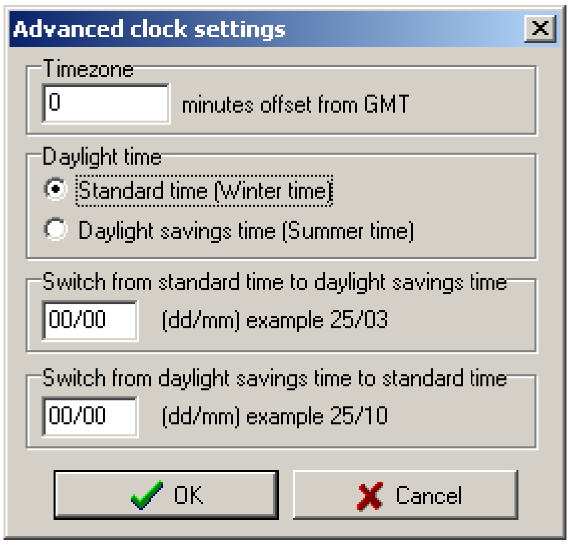
Synchronising the clock will only work with the latest models of E.D.&A. HPCs (February 2001).


Synchronize HPC-clock with PC-clock

HPC clock
Date: 01-02-2001
Time: 12:10:43
Standard time (winter time)

PC clock
Date: 01-02-2001
Time: 12:10:48
Standard time (winter time)

Use the Advanced settings button to enter the time zone (in minutes GMT) and the daylight savings time (summer/winter time)


Advanced clock settings

Timezone
0 minutes offset from GMT

Daylight time
[✓ Standard time (Winter time)]
Daylight savings time (Summer time)

Switch from standard time to daylight savings time
00/00 (dd/mm) example 25/03

Switch from daylight savings time to standard time
00/00 (dd/mm) example 25/10

The HPC system will not automatically switch from standard time to daylight savings time or from daylight savings time to standard time. You can use this flag to state if the clock time is standard time or daylight savings time. This flag also allows you to write some EPL-code that will change the time automatically.

6.7 The WINDOW menu

In this menu the user finds all the options that allow changing the location and the looks of the various windows and that allow you to activate certain windows.

6.7.1 WINDOW - NEW EDIT WINDOW

This menu choice allows you to open a new word processing window.

6.7.2 WINDOW - TILE HORIZONTALLY

Organizes all the open windows horizontally over the available space.

6.7.3 WINDOW - TILE VERTICALLY

Organizes all the open windows vertically over the available space.

6.7.4 WINDOW - CASCADE

Organizes all the open windows in a cascade behind each other.

6.7.5 WINDOW - SELECTOR

Opens a window that allows you to quickly switch between the active windows.

6.7.6 WINDOW - INFO

Goes to the info window.

6.7.7 WINDOW - REGISTERS

Goes to the register window. This window shows the actual values and changes in the register memory of the HPC.

6.7.8 WINDOW - COMPILE RESULTS

Selecting this option shows the user the results of the most recent translation task. Every line of this window shows us, from the left to the right, the name of the file in which the error was found and the line and the column of the error.
If you make use of the menu tasks PROJECT – NEXT ERROR and PROJECT – PREVIOUS ERROR, the content of the word processor is being synchronized with the error that is indicated at that moment in the compile results window.

6.7.9 WINDOW - BREAKPOINTS

Goes to the breakpoint window.

6.8 The HELP menu

This menu shows you the version number of EWB32 and where you can find E.D. &A. on the Internet.

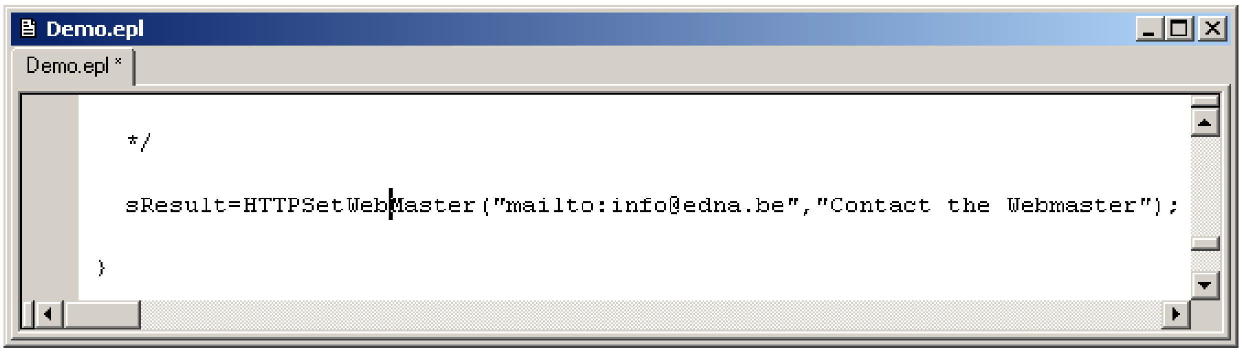
- E.D.&A. help documentation
- EWB32 version number
- Contact data of E.D.&A.

6.8.1 HELP - HELP

The HTML help documentation can be used in several ways in EWB.

6.8.1.1 Use F1 button

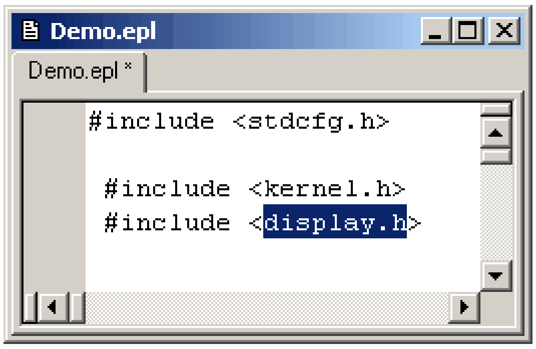
In the source code of the word processor the cursor can be placed on a word, for example a function.



```
*/
sResult=HTTPSetWebMaster("mailto:info@edna.be","Contact the Webmaster");
}
```

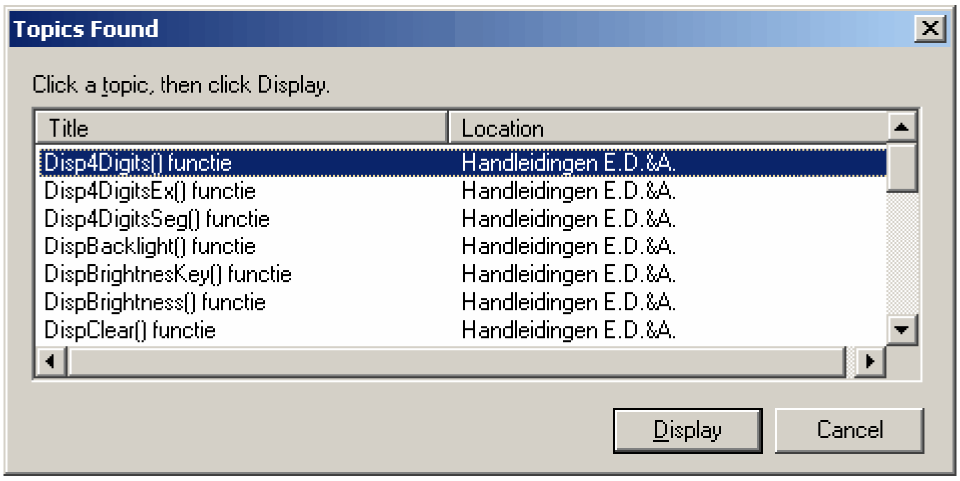
If at this moment F1 is pressed, the documentation about the function will be shown. If no direct information is found, a list with key words will be shown in the help file (the keyword that is most appropriate is marked). If a selection is been made in the source code of the word processor, this selection will be searched for.

Attention ! Most of the punctuation marks such as a dot or a comma are not taken into account in a word selection. If you indicate with your cursor an include name such as "KERNEL.H", the word KERNEL will automatically be selected and not the combination KERNEL.H (when pressing F1). If you want to select the combination, you first have to make a selection before you press F1.



```
#include <stdcfg.h>

 #include <kernel.h>
 #include <display.h>
```

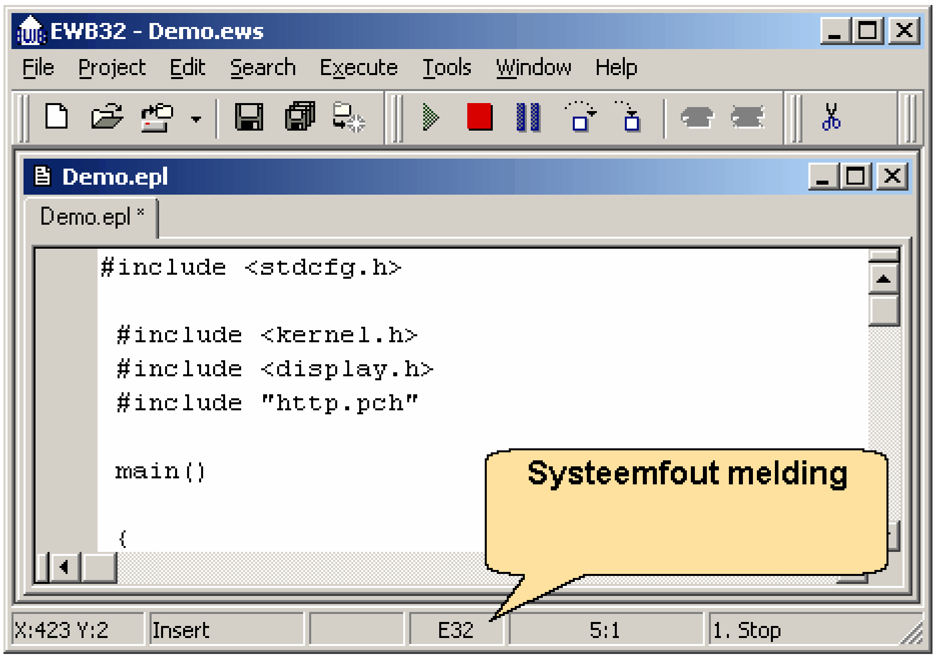
If you press F1 in this situation (after you have made a selection), you will see the following list:



Title	Location
Disp4Digits() functie	Handleidingen E.D.&A.
Disp4DigitsEx() functie	Handleidingen E.D.&A.
Disp4DigitsSeg() functie	Handleidingen E.D.&A.
DispBacklight() functie	Handleidingen E.D.&A.
DispBrightnesKey() functie	Handleidingen E.D.&A.
DispBrightness() functie	Handleidingen E.D.&A.
DispClear() functie	Handleidingen E.D.&A.

These are the functions of the display.h library.

6.8.1.2 Information : system error

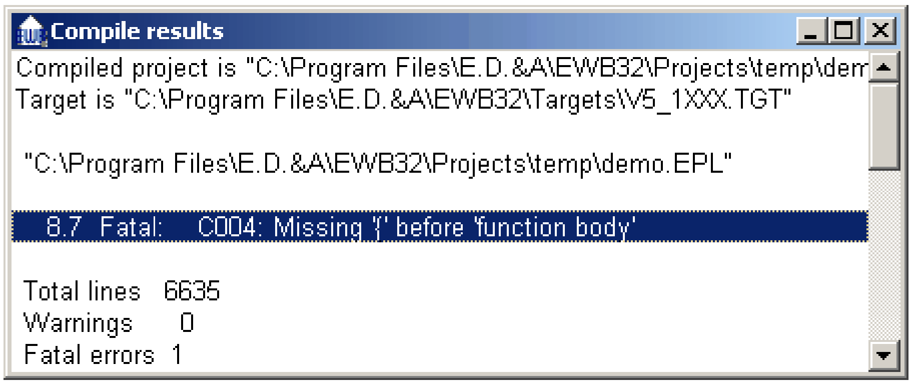


```
#include <stdcfg.h>

 #include <kernel.h>
 #include <display.h>
 #include "http.pch"

 main()

 {
```

```
X:423 Y:2   Insert   E32   5:1   1. Stop
```

In this example there is a system error E32. If you double-click E32 the available information about this system error appears.

6.8.1.3 Error messages of the translator



```
Compiled project is "C:\Program Files\E.D.&A\EWB32\Projects\temp\dem
Target is "C:\Program Files\E.D.&A\EWB32\Targets\V5_1XXX.TGT"

 "C:\Program Files\E.D.&A\EWB32\Projects\temp\demo.EPL"

   8.7  Fatal:    C004: Missing '{' before 'function body'

 Total lines   6635
 Warnings      0
 Fatal errors  1
```

If there is an error after compiling, the line with the error message can be selected in the results window. Then press F1 to receive the information about the error message.

6.8.2 HELP - ABOUT

Via this menu the version of EWB and data about E.D.&A. can be consulted.

7 Solve the problem

7.1 Technical support

Check whether the problem that has occurred is not described in this chapter. The guidelines that are given often lead to a quick solution of various problems. But if you do not succeed in solving your problems, we are here to help you. We do ask you to check a few things in advance and write some things down. This way we can work as efficiently as possible. The following information is important to write down:

- Which version of EWB32 do you use? You can find this by means of the menu task HELP-ABOUT.
- Which MS Windows operating system do you use? Which version is it?
- How much RAM memory do you have available?
- Is the problem reproducible or did it only occur once?
- Write down the exact message(s) that appear on the screen or try to describe exactly what does or does not happen. If possible, also write down the actions that have to be executed to reproduce the problem.
- If the problem occurs quite often during the translation of an HPC program, we would really appreciate it to receive an unchanged copy of your project so that we can isolate the problem as efficiently as possible.

7.2 Communication problems

7.2.1 No communication between EWB32 and the HPC (indication OFFLINE or NO NIF)

A communication problem between EWB32 and the connected HPC can have various causes. It is necessary to proceed methodically for finding a solution. Follow the following points step by step. They help you to find the cause!

- Check whether the HPC is live. Consult the Hardware Reference Manual of the HPC for the correct connections.

- Then check the connection cable between the network interface and the HPC. If you opted for a serial network interface on COM1 or COM2, then you have to make use of a programming cable of the type PP-80 to connect the PC with the HPC. Check whether you have not made any mistakes in case you have built the programming cable yourself. The guidelines for making a programming cable can be found in the Hardware Reference Manual of the HPC. Notice that the function of the pins on the DB-9 connection of the HPC does not correspond with the function of the pins on a DB-9 connection of a PC. Make sure that you select the correct serial port (COM1 or COM2) on the side of the PC and that you, on the side of the HPC, make use of the programming connection that is situated on the CPU card or on the basic appliance. If you make use of an ArcNet connection for the communication between the PC and the HPC, you have to use a coax cable type in order to connect the PC with the HPC. Consult the ArcNet Hardware Reference Manual for the correct settings of the ArcNet interface card.

- Check whether the settings for the test mode are correct. Check whether the network interface corresponds with the interface you wish to use to communicate with the HPC (serial versus ArcNet). Compare the setting of the 'Remote Node Number' with the address that is set on the HPC. Both addresses have to correspond. If this is not the case, communication between those two is impossible. Setting the address of an HPC is described in detail in the Hardware Reference Manual of the HPC. Consult the paragraph that contains an overview of the test mode settings.

- If there still is no communication between EWB32 and the HPC you have to check whether the yellow receiving LED on the HPC blinks regularly. If this is not the case, it means that:

1. There is not a (indication No NIF) serial network interface installed.
2. The serial network interface makes use of the wrong serial port.
3. The serial cable on the side of the PC is connected to the wrong serial port.
4. The serial cable on the side of the HPC is not connected to the programming port.
5. The serial cable that is used for the connection between the HPC and the PC is incorrect.
6. The programming port of the HPC is set on RS485 communication in stead of RS232.
7. The HPC is not live.
8. The serial port of the PC is defective.
9. The programming port of the HPC is defective.

- If the yellow receiving LED on the HPC does blink, you have to check whether the HPC reacts to the message it has received. You can see this by the red Led that lights up immediately after the yellow receiving LED has lightened up. If the red LED does not light up, this means that:

1. The address setting of the HPC does not correspond with the HPC Address setting in EWB.
2. The communication speed for the programming port does not correspond with the settings of the serial network interface. This is only applicable if the HPC has an adjustable communication speed for the programming port.
3. The serial port the HPC is not a programming port or that the programming port is not in programming mode. Activating the programming mode of the programming port happens automatically by placing the plug on the DB-9 connection. Check whether the electrical connections of the programming cable are correct, especially the bridge connections on the DB-9 connection for the HPC.
4. The serial cable that is used for the connection between the HPC and the PC is incorrect.
5. The serial port of the PC is defective.
6. The programming port of the HPC is defective.
7. The program that currently is sending data through the serial port to the HPC does not use the programming protocol, or in other words is not EWB32.

- If the red Led does light up, this means that:

1. The serial cable that is used for the connection between the HPC and the PC is incorrect.
2. The serial port of the PC is defective.

7.2.2 Communication error during the transfer of a program

If the error message Communication Error appears on the screen during the transfer of an HPC program from the PC to the HPC, after the FLASH Eprom has been erased, you have to go through the following steps:

- Select the menu task Project / options.
- In the dialog window Communication, column Network Interface.
- Check whether the correct interface has been selected.

7.2.3 The HPC stays in STOP mode after the menu task EXECUTE-RUN (F5)

- Check whether a program is loaded in the HPC. This is indicated by the RUN/HALT status indications or by the 7-segment display on the CPU module of the HPC. Consult the Hardware Reference Manual of the HPC for an overview of the status indications of the HPC concerned.
- Check whether there are active fatal system errors in the HPC. These are indicated by the HPC system error indicator on the status bar at the bottom of the screen in the Error code form. Consult chapter HPC system errors 81 for the correct meaning of the system errors.

7.3 EWB32 is suddenly closed off with a message on top of the screen

Write down the exact error message that appears on the screen and the circumstances in which the error occurs. Check whether the error is reproducible by repeating the action. Follow the guidelines in Technical support 55 . Contact E.D.& A. with the information you have gathered.

8 Messages of the translator

During the translation of an EPL program it is possible that the compiler generates messages concerning the course of the translation process. There are 2 kinds of messages. On the one hand there are the warnings and on the other hand the fatal errors. A warning indicates a possible problem that, however, can be dealt with by the compiler. A fatal error is a serious problem that cannot automatically be dealt with by the compiler. If there did not occur fatal errors during the translation process, the HPC program can be sent to the HPC. But we do advice you, the user, to carefully read the warnings and try to rectify them all.
The format of the messages in the 'Compile Results Window' is as follows:

```
Line, Column, Type, Error number : Error message
```

Line	Line number where the error occurred.
Column	Column where the error occurred.
Type	Type of error (Fatal or Warning)
Error number	Numerical indication of the error in the form Cnnn and nnn goes from 001 up to and including 999. Contact E.D.& A. if an error occurs which is indicated as Annn. This is an indication of a serious error during the translation and cannot be solved by the end user.
Error message	Textual description of the error.

8.1 Description of the various error messages

8.1.1 C001 Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name is valid and whether the file is not being used by another application.

8.1.2 C002: Unexpected end of file

The compiler is at the end of the program file but still expected one or more program lines. Check what causes the missing program lines.

8.1.3 C003: Syntax error : 'problem'

The compiler has found an error concerning the syntax of the EPL programming language. Check what causes this, using the problem description. Consult the EPL reference manual concerning the syntax of the EPL programming language.

8.1.4 C004: Missing 'symbol 1' before 'symbol 2'

The compiler expected 'symbol 1' before 'symbol 2'. Check why 'symbol 1' is missing. If necessary, add 'symbol 1' to the source program.

8.1.5 C005: Unmatched close comment

The compiler suspects that a comment block has not been closed off correctly by the symbol '*/'. Check whether this indeed is the case and, if necessary, correct the source program.

8.1.6 C006: Open compound statements (missing '}')

The compiler suspects that one or more compound blocks ({…}) have not been closed off correctly by the symbol '}'. Check whether this indeed is the case and, if necessary, correct the source program.

8.1.7 C007: Illegal function or declaration

The compiler expected the name of a function on the indicated place but the name that is found is not a valid function name. Check whether this location may follow a function indication. If this is not the case, this may indicate a faulty program structure in the previous lines, such as an extra '}' symbol. Check whether the signs in the function name are valid. Consult the EPL reference manual for an overview of the valid signs.

8.1.8 C008: Invalid type casting

The indication for a type casting is invalid. Consult the EPL reference manual for an overview of the valid type castings.

8.1.9 C009: Missing ';'

The compiler expected the symbol ';' on the indicated location, but could not find it. Check whether the symbol is only missing on the indicated location and whether the structure of the complete expression is valid.

8.1.10 C010: Invalid physical output

The indication for a digital or an analogue output is invalid. Check the spelling for a digital or an analogue output.

8.1.11 C011: Invalid target node selection for physical output

The indication of the HPC address for a digital output is invalid. Check the structure for the indication of a digital output.

8.1.12 C012: Invalid physical output card selection

The indication of the card number for a digital output is invalid. Check the structure for the indication of a digital output.

8.1.13 C013: Invalid physical output number

The indication of an output on a certain digital output card is invalid. Check the structure for the indication of a digital output.

8.1.14 C014: Invalid physical input

The indication for a digital or an analogue input is invalid. Check the spelling for a digital or an analogue input.

8.1.15 C015: Invalid target node selection for physical input

The indication of the HPC address for a digital input is invalid. Check the structure for the indication of a digital input.

8.1.16 C016: Invalid physical input card selection

The indication of the card number for a digital input is invalid. Check the structure for the indication of a digital input.

8.1.17 C017: Invalid physical input number

The indication of an input on a certain digital input card is invalid. Check the structure for the indication of a digital input.

8.1.18 C018: Invalid internal output

The indication for an internal auxiliary contact is invalid. Check the spelling for internal auxiliary contacts.

8.1.19 C019: Invalid target node selection internal output

The indication of the HPC address for an internal auxiliary contact is invalid. Check the structure for the indication of the internal auxiliary contacts.

8.1.20 C020: Invalid internal output number

The indication of an internal auxiliary contact is invalid. Check the structure for the indication of an internal auxiliary contact.

8.1.21 C021: Invalid word

The indication for an internal word register is invalid. Check the spelling for an internal word register.

8.1.22 C022: Invalid target node selection

The indication of the HPC address for an internal word register is invalid. Check the structure for the indication of the internal word registers.

8.1.23 C023: Invalid word number

The indication of an internal word register is invalid. Check the structure for the indication of an internal word register.

8.1.24 C024: Invalid argument

The string on the indicated location is not a valid name for a variable and also is not a valid constant.

8.1.25 C025: Identifier truncated : 'identifier'

The length of the symbolic name identifier is longer than allowed and it is being reduced by the compiler. Restrict the length of a symbolic name.

8.1.26 C026: New line character in string constant

The compiler discovered the beginning of a new program line when reading a string ("…"). The indication for the beginning and the end of a string has to occur on one and the same program line. It is, however, possible to tie together various separate strings if these strings are only separated mutually by spaces or by the beginning of a new line.

8.1.27 C027: New line character in character constant

The compiler discovered the beginning of a new program line when reading a character constant ('…'). The indication for the beginning and the end of a character constant has to occur on one and the same program line.

8.1.28 C028: Invalid binary escape character

The indication of a binary value in a character constant or a string (\0b…) contains invalid signs. A binary value has to consist of the numbers 0 and 1.

8.1.29 C029: Invalid hex escape character

The indication of a hexadecimal value in a character constant or a string (\0x…) contains invalid signs. A hexadecimal value has to consist of the numbers 0 up to and including 9 and the letters A up to and including F.

8.1.30 C030: Invalid escape character

The indication of a control character in a character constant or a string (\0x…) is invalid. Consult the EPL reference manual for an overview of the valid control characters.

8.1.31 C031: Unexpected end of file in comment session

The compiler has reached the end of the program file while it was going through a comment session. Check whether the beginning of the comment session is valid and why the indication at the end of the comment session is missing.

8.1.32 C032: Too many declarations

The maximum number of global and local variables is exceeded. This is the internal limit of the compiler. Contact E.D.& A. for a new version of the compiler.

8.1.33 C033: Undeclared identifier : 'identifier'

The compiler does not know the symbolic name 'identifier'. Check whether the indication is written correctly. If 'identifier' refers to a variable, you have to check whether this has actually been created. If 'identifier' refers to a function, you have to check whether this is declared.

8.1.34 C034: Illegal name for macro

The indication for a 'macro' or a symbolic name contains invalid signs. Consult the EPL reference manual for an overview of the valid signs.

8.1.35 C035: Too many macros

The maximum number of symbolic indications has been exceeded. This is the internal limit for the compiler. Contact E.D.& A. for a new version of the compiler.

8.1.36 C036: Macro pool full

The maximum memory capacity that has been reserved for the symbolic indications has been exceeded. This is the internal limit for the compiler. Contact E.D.& A. for a new version of the compiler.

8.1.37 C037: Line too long

The maximum length of a line in the Epl source program has been exceeded. Spread the task over one or more lines in the source program.

8.1.38 C038: Intertarget networking not supported

The name of a variable in the HPC is provided with an HPC network address. This is not being supported at the moment. You can only refer to own variables.

8.1.39 C039: Type 'float' not supported

A numerical constant was presented with a floating comma. Numbers with a floating comma are not yet being supported at the moment.

8.1.40 C040: Macro redefinition : 'macro name'

The symbolic indication 'macro name' is being redefined with another replacement. This is not allowed. Choose another name for the new symbolic indication.

8.1.41 C041: Pre-declared illegal

The pre-declared variable cannot be used in this context.

8.1.42 C042: Negative constant invalid

Only constants with a positive value are valid in this context.

8.1.43 C043: Missing ','

The compiler expected ',' but found another sign. Check why this sign is missing.

8.1.44 C044: Not a constant value

The value of an expression has to result in a constant number. This, however, is not the case.

8.1.45 C045: No matching #if

An #endif task or an #else task for the preparation is not preceded by an #if task. Check the structure of the program.

8.1.46 C046: Too many nested include files

The maximum number of #include tasks that can be nested is exceeded. Reduce the number of nested #include tasks. A nested #include task is an #include task that has to be activated before the previous #include task is totally completed.

8.1.47 C047: Illegal include filename

The indication for the file name of an #include task is invalid. Consult the EPL reference manual for the application of the “include task.

8.1.48 C048: Macro not defined : 'macro name'

The symbolic indication 'macro name' is not yet created or does not exist anymore.

8.1.49 C049: Message

General message that is generated from the EPL program during the translation. This message usually gives an additional indication concerning the translation of the source program.

8.1.50 C050: System configuration changes no longer valid

The translation of the program has reached the point where changes in the configuration of the HPC by means of the #config task are no longer possible. The configuration of the HPC has to occur at the beginning of the HPC program.

8.1.51 C051: System configuration redefinition

A configuration parameter is being redefined. This is not allowed. Remove one or both of the #config tasks.

8.1.52 C052: System configuration not yet completed

The configuration of the system is not yet completed. Complete the configuration of the system by means of the #config tasks before accepting the actual program line.

8.1.53 C053: Value out of range

The value of the configuration parameter is out of the valid range. Consult the EPL reference manual for an overview of the valid values for the #config task.

8.1.54 C054: Target out of data memory

The configuration of the data memory of the HPC exceeds 64 Kbyte (65536 bytes). This is not allowed. Reduce the required memory.

8.1.55 C055: Missing ':'

The compiler expected the symbol ':' but found another sign. Check the spelling of the #config task.

8.1.56 C056: Compiler limit - Too many I/O cards

The maximum number of input and output cards that is supported by the compiler has been exceeded. This is an internal limit for the compiler. Contact E.D.& A. for a new version of the compiler.

8.1.57 C057: Missing '.'

The compiler expected the symbol '.' but found another sign. Check the spelling of the #config task.

8.1.58 C058: IO-card not configured

In the EPL program there is a reference to a non-configured input card or output card. Check whether the indication in the program is correct. Make sure that the respective card is configured before it is referred.

8.1.59 C059: Physical output not available on specified card

A certain output is not configured on the specified card. Check whether the indication in the program is correct.

8.1.60 C060: Physical input not available on specified card

A certain input is not configured on the specified card. Check whether the indication in the program is correct.

8.1.61 C061: Internal word not available in current configuration

A certain internal memory register is not configured. Check whether the indication in the program is correct.

8.1.62 C062: Internal output not available in current configuration

A certain internal output is not configured. Check whether the indication in the program is correct.

8.1.63 C063: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause that is mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name is valid and/or whether the file is not being used by another application.

8.1.64 C064: Cannot write 'file name' : cause

The compiler cannot write the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file is not used by another application and whether the hard disk is not full.

8.1.65 C065: Left argument must be a left value

The left argument of an operator must indicate an element that can be used for saving data. The indication 'S1', e.g. is a 'left value'. The constant '37', e.g. is not a 'left value'.

8.1.66 C066: Precision lost in conversion of different integral types

During the type conversion of a word to a word with a smaller width, the most important bits will go lost because they cannot be saved. This may lead to miscalculations. If you, e.g. save a word of the long type in a word of the short type, this error notification will appear.

8.1.67 C067: 'signed' ignored

The indication 'signed' is superfluous and is ignored by the compiler.

8.1.68 C068: 'unsigned' ignored

The indication 'unsigned' is superfluous and is ignored by the compiler.

8.1.69 C069: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name is valid and/or whether the file is not being used by another application.

8.1.70 C070: Cannot read 'file name' :cause

The compiler cannot read the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file is not being used by another application.

8.1.71 C071: Cannot read 'file name' :cause

The compiler cannot read the indicated file because of an error mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file is not being used by another application.

8.1.72 C072: Cannot write 'file name' : cause

The compiler cannot write the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file is not being used by another application or whether the hard disk is not full.

8.1.73 C073: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name is valid and/or whether the file is not being used by another application.

8.1.74 C074: Statement too complex - Please simplify

The statement that is being translated at the moment is too complex for the compiler. Split up the statement in various parts and translate the program once again.

8.1.75 C076: Assignment within conditional expression

The statement within a conditional task (if, while) makes use of an equalization (=) as computation with the lowest priority. This may point to a type error. Possibly this equalization needs to be replaced by the conditional test on equalization (==).

8.1.76 C077: Conditional expression is constant

The statement within a conditional task (if, while) results in a constant number. This may point to a faulty analysis or an error during the insertion of a program line.

8.1.77 C078: Read-only argument modified

A variable that was created as 'read-only' is modified by a computation. The digital inputs are an example of variables that can only be read.

8.1.78 C079: Stack frame not empty

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.79 C080: Statement too complex - Please simplify

The statement that is being translated at the moment is too complex for the compiler. Split up the statement in various parts and translate the program once again.

8.1.80 C081: Operator not supported

A certain operator is currently not yet supported by the compiler. Contact E.D.& A. for a new version of the compiler.

8.1.81 C082: Operand not supported

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.82 C083: Data type error

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.83 C084: Statement too complex - Please simplify

The statement that is being translated at the moment is too complex for the compiler. Split up the statement in various parts and translate the program once again.

8.1.84 C085: No instruction list for operator

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.85 C086: No P-Code argument

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.86 C087: Instruction list invalid

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.87 C088: No left operand for binary operator

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.88 C089: Statement has no effect

The statement as it occurs in the EPL program has no effect and is therefore superfluous. This may point to a typing error.

8.1.89 C090: Function has too many actual parameters

During the call the function receives more parameters than initially anticipated. Check the correct spelling of the function call by means of the declaration of the function.

8.1.90 C092: Definition of 'identifier' hides previous definition

The definition of a local variable with the name 'identifier' hides a previous definition of a (global) variable with the same name. Make sure that local variables and global variables have different names.

8.1.91 C093: Expected integer kernel code

The keyword 'kernel' has to be followed by a constant of the short type. This was not the case. Consult the EPL reference manual for the use of the keyword 'kernel'.

8.1.92 C094: Type modifier 'kernel' invalid

The use of the keyword 'kernel' was incorrect within the context. Consult the EPL reference manual for the use of the keyword 'kernel'.

8.1.93 C095: Unmatched type specifiers

The variable type that was indicated during the declaration of the built-in variables does not correspond. Consult the EPL reference manual for the declaration of the built-in variables.

8.1.94 C096: Signed/Unsigned mismatch

A certain computation requires that both arguments are signed equally. This is not the case. Make sure that both are of the signed type or that both are of the unsigned type. You can accomplish this by using the type conversion or by declaring one of both arguments differently.

8.1.95 C097: Compiler out of memory

The compiler is out of memory. Make sure that your system is provided with more data memory.

8.1.96 C098: Integer constant divide by zero

During the computation of a constant value a division by 0 was found. The result of a division by 0 cannot be processed further on.

8.1.97 C099: Must be left value

The accompanying argument of a unary operator has to indicate an element that can be used for saving data. The indication 'S1' e.g. is a 'left value'. The constant '37' e.g. is not a 'left value'.

8.1.98 C101: Subscript on non-array

The argument on which the index operator ([…]) is applied is not an array or a pointer. Consult the EPL reference manual for the application of the index computation.

8.1.99 C102: Not a valid pointer

Variables of the type m (bool), c (char) and s (short) cannot be used as pointers. Use a variable of type l (long) for creating a pointer.

8.1.100 C103: Illegal indirection

The indirection computation (*) can only be applied on an argument of the pointer type with exception of the pointer type void *.

8.1.101 C104: Illegal index, indirection not allowed

The argument on which the index operator ([…]) is applied is not an array or a pointer. Consult the EPL reference manual for the application of the index computation.

8.1.102 C105: Bad left argument

In an arithmetical computation with pointers, the left argument has not got the correct type. Consult the EPL reference manual for an overview of the arithmetical computations that are possible with pointers.

8.1.103 C106: Bad right argument

In an arithmetical computation with pointers, the right argument has not got the correct type. Consult the EPL reference manual for an overview of the arithmetical computations that are possible with pointers.

8.1.104 C107: Different levels of indirection

A computation with 2 pointers requires that both pointers have the same level. Consult the EPL reference manual concerning the application of pointers.

8.1.105 C108: Indirection to different types

The computation with 2 pointers requires that both pointers are of the same type. Consult the EPL reference manual concerning the application of pointers.

8.1.106C109: Illegal break

The keyword 'break' cannot be used in this context. The use of this keyword is only allowed within a 'while' loop or within a 'switch'task.

8.1.107C110: Illegal continue

The keyword 'continue' cannot be used in this context. The use of this keyword is only allowed within a 'while' loop.

8.1.108C111: Addition of pointers illegal

The addition of two pointers is an invalid computation.

8.1.109C112: Pointer subtracted from non-pointer

If the right argument of a subtraction is a pointer, then the left argument also has to be a pointer. This was not the case.

8.1.110C113: Cannot access directly

The type conversion (direct) is only possible for physical inputs and outputs. Consult the EPL reference manual for the use of the (direct) type conversion.

8.1.111C115: Cannot read direct output

An output that has been converted to the type (direct) cannot be read. Only writing tasks are possible.

8.1.112C116: Cannot write direct input

An input that has been converted to the type (direct) cannot be written. Only reading tasks are possible.

8.1.113C117: Not a physical input or output

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.114C118: Not a function : 'identifier'

One tried to use 'identifier' as function name when calling a function. 'Identifier', however, is not a function.

8.1.115C119: 'remanent' uses unconfigured variables

The #config tasks tries to make non-configured variables non-volatile Only configured variables can be made 'non-volatile'.

8.1.116C120: Base type must be 'long' for cast to pointer

Only variables of the type 'long' can be cast to a pointer.

8.1.117 C121: Type modifier 'readonly' invalid

The keyword 'read-only' is invalid in this context. Consult the EPL reference manual for the application of the keyword 'read-only'.

8.1.118 C122: Function prototype parameter list different from body : parameter number

In the description of a function, the type of the parameter, indicated by a number, does not correspond with the type of the parameter in the declaration of the function. Check why both differ and make them consistent with each other.

8.1.119 C123: Missing function prototype : 'identifier'

A call from the function 'identifier' does not have a function declaration. Functions can only be called upon if they are declared beforehand. Consult the EPL reference manual concerning the declaration of functions.

8.1.120 C124: Function has too few actual parameters : 'identifier'

The function 'identifier' receives less parameters during the call than initially was anticipated. Check the correct spelling of the function call by means of the declaration of the function.

8.1.121 C125: Function has too many actual parameters : 'identifier'

The function 'identifier ' receives more parameters during the call than initially was anticipated. Check the correct spelling of the function call by means of the declaration of the function.

8.1.122 C126: Different types in function : parameter number

When calling a function, the parameter type indicated with a number does not correspond with the parameter type in the declaration of the function. Check why both differ and, if necessary, make use of a type conversion so that they correspond with each other.

8.1.123 C127: Function has no body : 'identifier'

The function description for the 'identifier' function is missing.

8.1.124 C128: Function already has a body : 'identifier'

The 'identifier' function already has a function description.

8.1.125 C129: Compiler limit - Literal pool full

The maximum memory space, reserved for the strings within a function description, was exceeded. This is an internal limit of the compiler. Reduce the number of strings and split up a larger function description in various smaller function descriptions.

8.1.126 C130: Expected integer callback code

The keyword 'callback' has to be followed by a constant of the short type. This was not the case. Consult the EPL reference manual for the use of the keyword 'callback'.

8.1.127 C131: Type modifier 'callback' invalid

The use of the keyword 'callback' was incorrect within the context. Consult the EPL reference manual for the use of the keyword 'callback'.

8.1.128 C132: Illegal declaration of local

The declaration of a local variable is formulated incorrectly. Consult the EPL reference manual for the declaration of local variables.

8.1.129 C133: Type modifier 'static' invalid

The use of the keyword 'static' was incorrect within the context. Consult the EPL reference manual for the use of the keyword 'static'.

8.1.130 C134: Array needs size

The declaration of an array has to be accompanied by the indication of a length for the array that differs from 0.

8.1.131 C135: Compiler limit - Out of local stack space

The local variables require too much stack space. Reduce the number of local variables within the function. Make sure that you give arrays with many elements the keyword 'static' if you wish to place them locally.

8.1.132 C136: Array too big

The total length of an array exceeds 65536 bytes. This is not possible. Reduce the number of elements in the array.

8.1.133 C138: Two's complement taken of unsigned

The negative computation (-) is applied on an unsigned element. This computation is meaningless because an unsigned element does not have a sign.

8.1.134 C139: Expression yields to constant

A pulse operator is applied on a constant value. The result of this computation is a constant and is meaningless.

8.1.135 C140: Invalid pre-processor command

The task for the pre-processor is invalid. Consult the EPL Reference Manual for an overview of the valid tasks for the preprocessor.

8.1.136 C141: Expected formal parameter list, not a type list

In the function description every parameter has to be provided with a parameter type and a symbolic name. When declaring a function is it enough to indicate the type of every parameter.

8.1.137 C142: Compiler limit - Too many function parameters

The maximum number of parameters that can be given in a function call has been exceeded. Reduce the number of parameters in the function call.

8.1.138 C145: Function function name - unreferenced formal parameter : parameter name

In a function description the parameter name is not used. Check whether it is meaningful to introduce it in the parameter list.

8.1.139 C146: Function function name - unreferenced local variable : identifier

In a function description the local variable identifier is not used. Check whether it is meaningful to enter this local variable.

8.1.140 C147: Pre-declared illegal as parameter

The name of a pre-declared variable cannot be used as the name of a function parameter.

8.1.141 C148: Pre-declared illegal as local variable

The name of a pre-declared variable cannot be used as the name of a local variable.

8.1.142 C149: Empty character constant

The content of the character constant ('…') is void. This is not allowed.

8.1.143 C150: Truncation of constant value

The value of a constant is too large to be saved in the anticipated type. The compiler breaks off the value. This may lead to faulty results.

8.1.144 C151: Logical operation on address of string constant

Testing the equality of two strings cannot be realized by making use of the comparison computation (==). You have to make use of a series of functions that can be found in the 'string.h'library.

8.1.145 C152: Warning treated as fatal error - no output file generated

The compiler informs you that non-fatal errors will be treated as fatal errors. This is a setting the user can make in the option menu of the compiler.

8.1.146 C153: Local name already used as function name : identifier

The name of a local variable is already being used as the name of a function. Choose another name for the local variable.

8.1.147 C154: Local name already used as kernel name : identifier

The name of a local variable is already being used as the name of a 'kernel' variable. Choose another name for the local variable.

8.1.148 C155: Type 'void' illegal : identifier

The void type is not a valid basic type for the execution of the computation. Check why the basic type is void.

8.1.149 C156: Unknown size (base type void)

The compiler cannot determine the size of an element of which the type is void. Check why the type is void. If necessary, make use of a type converter to change the basic type.

8.1.150 C157: Left operand has type 'void'

The left element has the void type. Check why the type is void and correct it if necessary.

8.1.151 C158: Right operand has type 'void

The right element has the void type. Check why the type is void and correct it if necessary.

8.1.152 C159: Operand has type 'void'

The argument for a unary computation has the void type. Check why the type is void and correct it if necessary.

8.1.153 C160: Parameter has type 'void' : parameter number

The function parameter has the void type. Check why the type is void and correct it, if necessary.

8.1.154 C161: Cast of void term to non-void

It is impossible to cast the void type to a non-void type.

8.1.155 C162: Controlling expression has type 'void'

The result of an expression in if, while or switch task is of the void type. This is not allowed.

8.1.156 C163: Function must return a value : function name

A function description does not make use of the return task to return a value to the called function. According to the declaration of the function, this has to occur. Make sure that the function returns a value by making use of the return task, or change the return value of the function in void when declaring the function.

8.1.157 C164: 'void' function returning a value

A function description makes use of the return task to return a value to the called function. According to the declaration of the function this does not happen. Make sure that the function does not return a value by making use of the return task, or change the return value of the function from void to the suited type when declaring the function.

8.1.158C165: 'return' : Different levels of indirection

The level of the pointer in the return value of a function does not correspond with the level of the pointer when declaring the function. Consult the EPL reference manual concerning the application of pointers.

8.1.159C166: 'return' : Indirection to different types

The type of the pointer in the return value of a function does not correspond with the type of the pointer when declaring the function. Consult the EPL reference manual concerning the application of pointers.

8.1.160C167: Function prototype has void parameter list : function name

The function 'function name' receives parameters during the call. When declaring a function, however, it was specified that there were no parameters. Check whether the spelling of the function call is correct by means of the declaration of the function.

8.1.161C168: Keyword 'remote' invalid

The keyword 'remote' is invalid within the context. Consult the EPL reference manual for the application of the keyword 'remote'.

8.1.162C169: Cannot access remote i/o directl

The type conversion (direct) cannot be applied on 'remote' inputs and outputs.

8.1.163C170: Address-of operator on string constant ignored

The address-of computation (&) is ignored if it is applied on a string.

8.1.164C171: Address-of operator on function ignored

The address-of computation (&) is ignored if it is applied on the name of a function.

8.1.165C172: Address-of operator on array ignored

The address-of computation (&) is ignored if it is applied on the name of an array.

8.1.166C173: Compiler limit - Too many levels of indirection

The maximum number of levels for the pointer is exceeded. Change your expression so that the level of the pointer reduces.

8.1.167C175: Missing '#endif'

The compiler finds one or more #endif tasks at the end of an EPL program. Check the course of the program to see where these are missing.

8.1.168C176: Fields stack overflow

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.169 C177: Fields stack underflow

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.170 C178: Fields stack not empty

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.171 C179: Cannot initialize variable

The compiler cannot statically initialize this type of variable. Consult the EPL reference manual for an overview of the initialization of variables.

8.1.172 C180: Infinite optimizer loop

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.173 C181: Too many initializers

The list of initialization values for an array is longer than the number of elements in the array. This is impossible. The number of initialization values has to correspond with the length of the array.

8.1.174 C182: Not enough initializers

The list of initialization values for an array is shorter than the number of elements in the array. This is impossible. The number of initialization values has to correspond with the length of the array.

8.1.175 C183: Compiler limit - Initialized data pool full

The maximum memory space for the saving of initialization values is exceeded. Reduce the number of initialization values in the program.

8.1.176 C184: Cannot install FPMATH exception handler

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.177 C185: Null statement used

The compiler found a statement that only consists of the symbol for the end of a statement (;). This may point to a typing error.

8.1.178 C186: Too many remanent blocks

The maximum number of non-volatile blocks in a program is exceeded. Reduce the number of blocks you wish to keep non-volatile by removing blocks or by adding 2 blocks together.

8.1.179 C187: Type modifier 'remanent' invalid

The use of the keyword 'remanent' was incorrect within the context. Consult the EPL reference manual for the use of the keyword 'remanent'.

8.1.180 C188: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name exists and/or whether the file is not being used by another application.

8.1.181 C189: Cannot write 'file name' : cause

The compiler cannot write the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file name exists and/or whether the file is not being used by another application or whether the hard disk is not full.

8.1.182 C190: Compiler limit - Too many remanent blocks

The maximum number of non-volatile blocks in a program is exceeded. Reduce the number of blocks you wish to keep non-volatile by removing blocks or by adding two blocks together.

8.1.183 C191: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system such as an invalid file name or a station that does not exist. Check whether the file name exists and/or whether the file is not being used by another application.

8.1.184 C192: Cannot read 'file name' : cause

The compiler cannot read the indicated file because of a cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file name exists and/or whether the file is not being used by another application.

8.1.185 C193: Target redefinition

The #target task is executed once more but with modified parameters. This is not allowed.

8.1.186 C194: Target already loaded : target specification

The #target task is executed at the moment that the target data for a certain target are already loaded. Changes are no longer possible.

8.1.187 C196: Cannot load TDC 'file name' : cause

A target dependent compiler module cannot be loaded in the data memory of the PC because of a cause mentioned in the error notification. This error can be caused by a faulty or incomplete installation of EWB32 or by a lack of data memory.

8.1.188 C197: Cannot load TDA 'file name' : cause

A target dependent compiler module cannot be loaded in the data memory of the PC because of a cause mentioned in the error notification. This error can be caused by a faulty or incomplete installation of EWB32 or by a lack of data memory.

8.1.189 C198: Illegal target filename

The indication for the file name after the #target task is incorrect. Consult the EPL reference manual for the correct composition of the #target task.

8.1.190 C199: Current target doesn't support direct access for the selected i/o type

The (direct) type converter cannot be applied on the input or the output because of the fact that the selected target does not support this. Consult the hardware reference manual of the target CPU for an overview of the support of the (direct) type conversion.

8.1.191 C200: Cannot unload TDC 'file name' : cause

A target dependent compiler module cannot be removed from the data memory of the PC because of the cause mentioned in the error notification.

8.1.192 C201: Cannot unload TDA 'file name' : cause

A target dependent compiler module cannot be removed from the data memory of the PC because of the cause mentioned in the error notification.

8.1.193 C202: Illegal use of reference ignored

The use of the reference operator during the creation of variables is not supported.

8.1.194 C203: Function parameter passing by reference requires left value

To give a function parameter by using the memory location where it is saved, it is necessary that the parameter is a so-called 'left value' .

8.1.195 C205: Only static locals can be initialized

To initialize local variables during the creation, it is necessary that these are of the 'static' type. Non-static variables cannot be initialized during the creation.

8.1.196 C206: Cannot access array element directly

Variables that are part of an array cannot be accessed directly. If you do wish to access the content of this variable, you can do this by taking the basic variable of the array and index it with a constant value that corresponds with the variable you wish to access.

8.1.197 C207: Only static locals can be made remanent

If you wish to make the local variables 'non-volatile', these have to be of the 'static' type. Non-static variables cannot be made 'non-volatile'.

8.1.198C208: More than one default for switch

In a switch task, the key word default occurs more than once. That is not allowed. Consult the EPL reference manual for the use of the switch task.

8.1.199C209: Compiler limit - Too many active cases

The maximum number of 'case' key words that the compiler can deal with has been exceeded. Reduce the number of 'case' keywords that has to be dealt with during the translation. If there is a switch task within another switch task, then you can simplify this by inserting the program lines in the outer switch task in a separate function.

8.1.200C210: Case requires constant of type 'short'

The constant value that follows the keyword 'case' has to be of the short type.

8.1.201C211: Switch expression must have type 'short'

The expression that follows the switch task has to result in a value of the short type.

8.1.202C212: Case value already used

For one and the same switch task, the same constant value is repeatedly added to a case keyword. This is impossible. Every value after a case keyword has to be unique for one and the same switch task.

8.1.203C213: Illegal default (outside switch)

The keyword default can only be used within a switch task.

8.1.204C214: Return of a 'void' expression

The value that follows the keyword return is void. That is not allowed.

8.1.205C215: Cast of non-void term to void

It is impossible to cast a void term to a non-void term.

8.1.206C216: Definition of identifier 1 conflicts with previous definition of identifier 2

The definition of a global variable identifier 1 conflicts with the definition of a global variable identifier 2. Change one definition.

8.1.207C217: System 'callback' function called by user program

The user program calls a function with the indication 'callback'. This is not normal and it may lead to serious errors during the execution of the program.

8.1.208C218: Current target doesn't support remote i/o

The target that is currently selected does not support 'remote' inputs or outputs.

8.1.209C219: Integer constant overflow

The result of a computation on constants is too large to be saved in a long. This leads to a loss of data. Check why the result of the computation cannot be saved in a long.

8.1.210C220: No matching POP Pcode for PUSH Pcode

This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

8.1.211C221: Unknown compiler directive

The compiler directive for the compiler (#pragma) could not be recognized. Consult the EPL reference manual for an overview of the valid compiler directive for the compiler.

8.1.212C222: Invalid option for compiler directive 'debug'

The parameter for the compiler directive '#pragma debug' is not valid. Consult the EPL reference manual for an overview of the valid tasks for the compiler directive '#pragma debug'.

8.1.213C223: Illegal case (outside switch)

The key word case can only be used within a switch task.

8.1.214C224: Invalid option for compiler directive 'mathoverflow'

The parameter for the compiler directive '#pragma mathoverflow' is not valid. Consult the EPL reference manual for an overview of the valid tasks for the compiler directive'#pragma mathoverflow'.

8.1.215C225: Cannot open 'file name' : cause

The compiler cannot open the indicated file because of the cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system, such as an invalid file name or a station that does not exist. Check whether the file name is valid and/or the file is not being used by another application.

8.1.216C226: Cannot write 'file name' : cause

The compiler cannot write the indicated file because of the cause mentioned after the name of the file. Mostly this refers to an error concerning the operating system. Check whether the file is not being used by another application and check whether the hard disk is not full.

8.1.217C227: Invalid macro name in default defines

The name of a macro that is specified in the option dialog window of the compiler is invalid.

8.1.218C228: Macro redefinition in default defines

A macro is being redefined in the option dialog window of the compiler.

8.1.219 C229: Missing ';' in default defines

The compiler expected the symbol ',' in the list with macros in the option dialog window of the compiler but found another sign.

8.1.220 C230: Type modifier 'automatic' invalid

The use of the key word 'automatic' was incorrect within the context. Consult the EPL reference manual for the use of the key word 'automatic'.

8.1.221 C231: No target specified

There is no target indication indicated for which the EPL program has to be translated. The compiler can only continue its work if a target HPC is specified. Consult chapter Default target 14 concerning the indication of a target indication. Consult the EPL reference manual for the indication of a target indication in the EPL program.

8.1.222 C232: Invalid option for compiler directive 'boundscheck'

The parameter for the compiler directive '#pragma mathoverflow' is invalid. Consult the EPL reference manual for an overview of the valid tasks for the compiler directive '#pragma mathoverflow'.

8.1.223 C233: Output code buffer overrun

Internal compiler buffer is too small. Please contact E.D.&A. for further support.

8.1.224 C234: Suspicious use of a reserved keyword : <keyword>

The use of a reserved key word can cause problems during the compiling process.

8.1.225 C235: Nested comments no longer supported - use #if 0 and endif instead");

Nested comments are no longer supported. Instead of this, please use the pre-treated tasks #if/#endif.

8.1.226 C999: Unknown error (number)

An unknown error has occurred during the translation of the program. This is a serious internal error in the compiler that cannot be solved by the user. Write down the circumstances in which the error occurs and contact E.D.& A. for a new version of the compiler.

9 HPC system errors

9.1 Overview

System errors can occur after the transfer and before the execution of a new program to an HPC or during the execution of a program on an HPC. There are two types of system errors, namely fatal system errors and non-fatal system errors.

The consequences of non-fatal errors are that:

- A notification is placed in the error system of the HPC.
- The execution of the HPC program continues.
- They are not mentioned on the status display of the HPC (if it is equipped with a display).

The consequences of fatal errors are that:

- A notification is placed in the error system of the HPC.
- The execution of the HPC program is interrupted. The HPC is put in STOP mode. The execution of the HPC program can only be continued if the cause of the fatal error is dealt with.
- The error code is represented on the status display of the HPC (if this is equipped with a display).

An error code always consists of a number and optionally one or more error parameters. These parameters give additional information concerning the error that occurred.

The error codes are divided in groups as follows:

E01…E19	global errors
E20…E29	errors concerning the input and output cards
E30…E39	errors concerning the structure of the HPC program or errors during the loading of a program
E40…E59	errors that occur during the execution of the program.

9.2 Explanation of the error codes

9.2.1 E01

Type
Fatal

Meaning
There is insufficient RAM memory available for executing the program.

Extra parameters
“Last error var1” indicates in which module the error occurred. For an overview of the various modules we refer to Additional error codes 91.

Solving the problem
Reduce the required memory by adapting the configuration. (#config pretreatment) of the HPC or install a RAM memory extension.

9.2.2 E02

Type
Fatal

Meaning
The bus cutoff at the end of the bus is missing.

Extra parameters
None

Solving the problem
Consult the Hardware Reference Manual of the HPC concerned about the placement of the bus cutoff.

9.2.3 E03

Type
Fatal

Meaning
A request for a program interruption has appeared in the HPC, but its cause could not be found. It is also possible that an invalid instruction was executed by the HPC.

Extra parameters
None

Solving the problem
Contact E.D.& A. for further advice.

9.2.4 E04

Type
Fatal

Meaning
Two or more HPCs in a network have received the same HPC address.

Extra parameters
None

Solving the problem
Make sure that all the participants of a network have a unique address. Consult the Hardware Reference Manual of the HPC concerning the setting of the HPC address.

9.2.5 E05

Type
Fatal

Meaning
The CPU has read an invalid instruction.

Extra parameters
These 2 parameters show the segment and the offset where the fault has appeared.

Solve the problem
This can indicate a hidden problem in your EPL program. Please check if your program use pointers in a correct way. It is possible to use the source code debugger option "array index validation" to localize the cause of the problem.

9.2.6 E06

Type
Fatal

Meaning
During the start up the eeprom could not be read.

Extra parameters
None

Solve the problem
Please contact E.D.&A.

9.2.7 E08

Type
Fatal

Meaning
The HPC program contains a configuration that differs from the current configuration of the non-transitory RAM memory. The data memory of the HPC cannot be configured again for the new program without the non-transitory data going lost.

Extra parameters
None

Solving the problem
This error status can be solved by executing a "cold" start or by loading another HPC program that does contain a corresponding configuration. For executing a cold start, we would refer to the menu task EXECUTE -INIT REMANENT 45.

9.2.8 E09

Type
Fatal

Meaning
Because of a serious system interference the content of the non-transitory memory has gone lost.

Extra parameters

None

Solving the problem
This error may be caused by an incorrect functioning HPC or by a serious error in the EPL application program. Verify the correct use of pointers in the EPL program. Contact E.D.& A. if the problem keeps occurring without an apparent reason.
Executing a "cold" start can only solve the error status. To execute a cold start, we would refer to the menu task EXECUTE -INIT REMANENT 45.

9.2.9 E10

Type
Not fatal

Meaning
There is a fault concerning the TCP/IP stack.

Extra parameters
None

Solve the problem
Use the HPCIPCFG program to make a further diagnose.
The most common causes are:

- TCP/IP stack is not installed in the HPC. You can check this by asking for the version number of the TCP/IP stack in the info window in EWB.
- The TCP/IP stack is installed so that it uses an Ethernet card. If this card is not installed in the HPC, this will cause an error.
- At least one IP setting is not complete or has come into conflict with the system set-up.

9.2.10 E20

Type
Non-fatal

Meaning
There is an extension card in the HPC that is not recognized by the system.

Extra parameters
"Last error var1" indicates the number of the extension key in which the unknown card is situated. Consult the Hardware Reference Manual for the correct numbering of the extension keys.

Solving the problem
Install a new version of the system monitor in the appliance. Contact E.D.& A. for receiving a recent version of the system monitor.

9.2.11 E21

Type
Non-fatal

Meaning
There is an extension card in the HPC that is not recognized by the system.

Extra parameters
"Last error var1" indicates the number of the extension key in which the unknown card is situated. Consult the Hardware Reference Manual for the correct numbering of the extension keys.

Solving the problem
Install a new version of the system monitor in the appliance. Contact E.D.& A. for receiving a recent version of the system monitor.

9.2.12 E22

Type
Fatal

Meaning
The set card number on one of the extension cards is invalid for this HPC model.

Extra parameters
“Last error var1” indicates the number of the extension key in which the faulty set card is situated.

Solving the problem
Set the address of the specified card on a valid value. Consult the Hardware Reference Manual for setting the card number and for a list of valid card numbers.

9.2.13 E23

Type
Fatal

Meaning
Two or more cards of the same type are set on the same card address.

Extra parameters
“Last error var1” and “Last error var2” indicate in which extension keys these cards are situated.

Solving the problem
Choose a unique address for one type of card. Consult the Hardware Reference Manual for setting the card number and for a list of valid card numbers.

9.2.14 E24

Type
Fatal

Meaning
The HPC could not find an extension card in the system, while the HPC program requires this.

Extra parameters
“Last error var1” and “Last error var2” indicate in which extension keys these cards are situated.

Solving the problem
This error may occur when the HPC program was not configured as "tolerant", but as "inclusive" or "exclusive". Consult the EPL reference manual concerning the use of the tasks #config tolerant, #config inclusive and #config exclusive. Check whether the real configuration of the inputs and outputs of the system corresponds with the software configuration of the system.

9.2.15 E30

Type
Fatal

Meaning
During the validation of the loaded HPC program a fatal problem was found.

Extra parameters
"Last error var1" indicates the cause of the problem. For an overview of the possible causes we would like to refer to Additional error codes 91

Solving the problem
Translate the project and send it once more to the HPC. Consult E.D.& A. if the problem keeps occurring.

9.2.16 E31

Type
Fatal

Meaning
The HPC program requires the use of a certain call back function that is not supported by the HPC. This may occur when a new HPC program is installed on an older HPC.

Extra parameters
“Last error var1” indicates the reference number of the desired call back function.

Solving the problem
Install a new version of the system monitor in the appliance. Contact E.D.& A. for receiving a recent version of the system monitor.

9.2.17 E32

Type
Fatal

Meaning
The HPC program requires the use of a certain kernel function that is not supported by the HPC. This may occur when a new HPC program is installed on an older HPC.

Extra parameters
“Last error var1” indicates the reference number of the desired kernel function.

Solving the problem
Install a new version of the system monitor in the appliance. Contact E.D.& A. for receiving a recent version of the system monitor.

9.2.18 E33

Type
Fatal

Meaning
The HPC program divides the memory of the HPC in too many different non-volatile fields.

Extra parameters
"Last error var1" indicates the reference number of the desired call back function.

Solving the problem
Make sure that the number of various non-volatile blocks is not bigger than 64. Try to build the non-volatile blocks in a way that they consist of successive non-volatile variables so that the HPC can process them as one block.

9.2.19 E40

Type
Fatal

Meaning
The HPC program requires the use of a certain kernel function or kernel variable that is not supported by the HPC. This may occur when a new HPC program is placed on an older HPC.

Extra parameters
“Last error var1” indicates the reference number of the required kernel function or kernel variable.

Solving the problem
Install a new version of the system monitor in the HPC. Contact E.D.& A. for receiving a recent version of the system monitor.

9.2.20 E41

Type
Non-fatal

Meaning
One of the parameters given to a kernel function call is invalid.

Extra parameters
"Last error var1" indicates which function argument is wrong.

Solving the problem
Consult the Reference Manual Library Functions for an overview of the kernel functions and their parameters.

9.2.21 E45

Type
Non-fatal

Meaning
The HPC program tries to install an option for which there has not been found a corresponding option key.

Extra parameters
"Last error var1" indicates in which module the error occurred.

Solving the problem
Install the required option keys. Consult the Hardware Reference Manual of the HPC for installing the option keys.

9.2.22 E46

Type
Non-fatal

Meaning
A kernel function could not found the desired input/output card.

Extra parameters
“Last error var1” indicates the card type and “Last error var2” indicates the card number.

Solving the problem
Check whether the function parameters are filled in correctly and whether the specified input/output card is installed in the HPC.

9.2.23 E47

Type
Non-fatal

Meaning
A kernel function could not find the desired serial port in the HPC.

Extra parameters
“Last error var1” indicates the number of the serial port.

Solving the problem
Check whether the function parameters are filled in correctly and whether the specified serial port is installed in the HPC.

9.2.24 E48

Type
Non-fatal

Meaning
A kernel function could not find the Real Time Clock (RTC) in the HPC.

Extra parameters
None

Solving the problem
Install the optional Real Time Clock in the HPC. Consult the Hardware Reference Manual of the HPC for guidelines concerning the installation of the Real Time Clock.

9.2.25 E49

Type
Non-fatal

Meaning
The HPC program contains non-volatile variables, but in the HPC there is no data memory with battery backup installed.

Extra parameters
None.

Solving the problem

Install a data-RAM with battery backup. Consult the Hardware Reference Manual of the HPC for the guidelines concerning the installation of a RAM with battery backup.

9.2.26 E50

Type
Fatal

Meaning
The HPC program has exceeded the maximum allowed execution time(300 ms) and has been interrupted by the system.

Extra parameters
None

Solving the problem
Check whether the HPC program does not contain any loops (while() task) of which the execution time is too long. If EWB32 is equipped with the source code debugger, it will indicate the program line the HPC was executing at the moment that the program cycle duration was exceeded by means of a red bar on the screen.

9.2.27 E51

Type
Fatal

Meaning
The HPC program has made an arithmetical computation in which the result has become too large to be saved in the type (char, short or long) of the arguments concerned. The consequence of this is that the result will no longer corresponds with the real value.

Extra parameters
None

Solving the problem
Check why the result of the computation becomes too large. If necessary, make use of a type of variable that has a larger width. If EWB32 is equipped with the source code debugger, it will indicate the program line where the error occurs, by means of a red balk on the screen.

9.2.28 E52

Type
Fatal

Meaning
The HPC program uses a pointer with the 0 or NULL value. This possibly means that a variable of the 'pointer to' type is not initialized before it is used.

Extra parameters
None

Solving the problem
Check whether the HPC program uses uninialized or incorrectly initialized pointers.

9.2.29 E53

Type
Fatal

Meaning
The HPC has executed an arithmetical division in which the divisor is equal to 0. The result of this computation is indefinite.

Extra parameters
None

Solving the problem
Check why the divisor of the division is 0. If EWB32 is equipped with the source code debugger, it will indicate the program line where the error occurs by means of a red bar on the screen.

9.2.30 E54

Type
Fatal

Meaning
The index of an array is bigger than or equals to the maximum number of elements reserved for this array. The value that is referred to is not an element of the indicated array.

Extra parameters
None

Solving the problem
Check why the array index exceeds the valid reach. If EWB32 is equipped with the source code debugger, it will indicate the program line where the error occurs by means of the red bar on the screen.

9.2.31 E55

Type
Non-fatal

Meaning
The HPC program sends data more quickly to the serial port than the port can send them.

Extra parameters
None

Solving the problem
Reduce the amount of data the serial port has to send or increase the transfer speed.

9.2.32 E56

Type
Non-fatal

Meaning
A serial port has received data more quickly than the HPC can process them.

Extra parameters
None

Solving the problem
Decrease the download speed of the serial port or make use of the handshake signals of the serial port for guiding the data transfer.

9.3 Additional error codes

Causes of invalid program files

Code	Description of the cause
0	The control sum of the program was incorrect.
1	The version number of the program heading is incorrect.
2	The loading address of the program is incorrect.

Module codes and description

Code	Description of the module
1	Basic functionality
5	Analog I/O System
10	Serial communication module
11	VNET communication module
15	Remote input/output module
20	RFUP module
21	VFUP module

10 Versions book 34

Version number	**Date**	**Adjustments**
1.0.01		Initial release of version 1
1.0.02	17/11/2003	Supplement to Source code word processor 24 en the edit mode 30
1.0.03	19/02/2004	Modification TOOLS - Synchronise clock 50
1.0.04	15/07/2004	Complete check manual

Index